

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☒ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

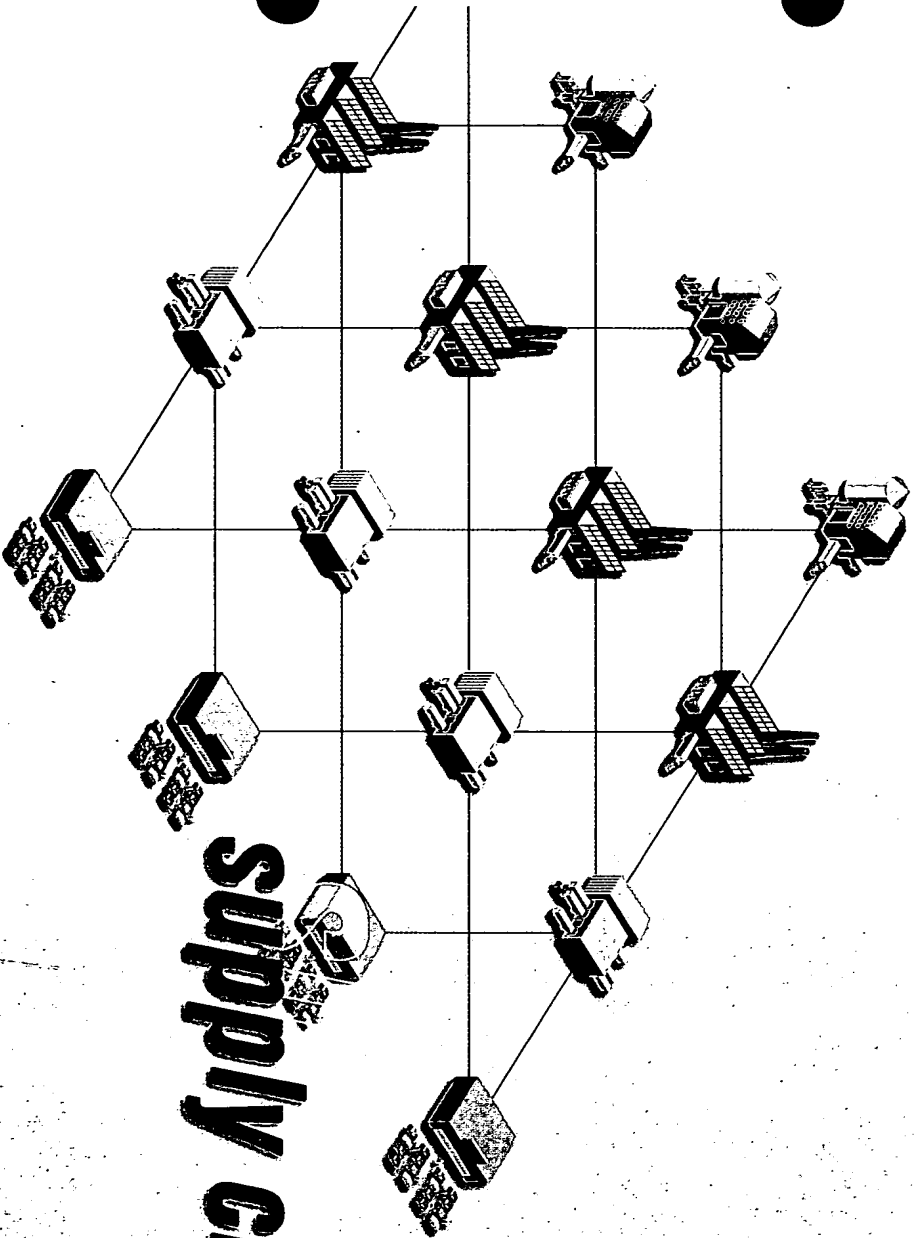
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.



RHYTHM®

MODEL REFERENCE MANUAL

intelligent decision-support solutions for supply chain planning and optimization



Supply chain planner

i2 Technologies *i2*



RHYTHM Supply Chain Planner Model Reference Manual

Copyright © 1998
i2 Technologies, Inc.
All rights reserved

This notice is intended as a precaution against inadvertent publication and does not imply publication or any waiver of confidentiality. The year included in the foregoing notice is the year of creation of the work.

Information in this document is subject to change without notice and does not represent a commitment on the part of i2 Technologies. The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of the agreement. It is against the law to copy the software on any medium except as specifically allowed in the license or nondisclosure agreement. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage or retrieval systems, for any purpose other than the purchaser's personal use without the express written permission of i2 Technologies.

The information and/or drawings set forth in this document and all rights in and to disclosing or employing the materials, methods, or techniques described herein are the exclusive property of i2 Technologies, Inc.

Unless otherwise noted, all names of companies, products, street addresses, and persons contained herein are part of a completely fictitious scenario or scenarios and are designed solely to document the use of an i2 Technologies product.

© 1998 i2 Technologies, Inc. All rights reserved. Printed in the United States of America. No part of this document may be reproduced in any form, by phoscopy, microfilm, xerography, or any other means, or incorporated into any information retrieval system, electronic or mechanical, without the written permission of i2 Technologies, Inc.

The X Window System is a trademark of the Massachusetts Institute of Technology.

UNIX and Unix are registered trademarks of AT&T.

OIL, Constraint, Anchored Optimization, CAO, and RHYTHM are trademarks of i2 Technologies, Inc.

i2 logo and RHYTHM logo are trademarks of i2 Technologies, Inc.

This manual was written, illustrated, and produced with the X/Motif and Windows NT FrameMaker document publishing software.

Written by the development group of i2 Technologies, Inc.

Version 3.07

i2 TECHNOLOGIES, INC.
909 East Las Colinas Blvd.
16th Floor
Irving, Texas 75039
USA

March 26, 1998

Revision History

<u>Edition</u>	<u>Date</u>	<u>Reason for Revision</u>
3.04Beta	09/16/96	Beta Release
3.06A	10/14/96	Production Release
3.05Beta	12/20/96	Beta Release
3.05A	01/31/97	Production Release
3.06Beta	10/20/97	Beta Release
3.06A	11/10/97	Production Release
3.07Beta	03/03/98	Beta Release
3.07A	03/27/98	Production Release

Table of Contents

1	Introduction	25
1.1	Preface	25
1.2	Overview	25
1.3	RHYTHM Manuals	26
1.4	Modeling	27
1.4.1	Expressions	27
1.4.2	Types	27
1.4.3	Fields	27
1.4.4	Models	28
1.4.5	Submodels	28
1.4.6	Extensions	29
1.4.7	User-Defined Fields	29
1.5	Manual Organization	30
1.5.1	HyperText	30
1.5.2	Notation	30
1.5.3	Chapters	30
1.5.4	Fields	30
1.5.5	Models	30
1.5.6	Exceptions	31
2	Basic Types	33
2.1	Cell State	33
2.2	Computed String	33
2.3	Date	33
2.4	Date Range	34
2.5	Event	34
2.6	Expression	34
2.7	Func	34
2.8	Horizon Date	34
2.9	ID	35
2.10	Integer	35
2.11	Logical	35
2.12	Measure	35
2.13	Measure Unit	35
2.14	Money	36
2.15	Number	37
2.16	Pathname	37
2.17	Percentage	37
2.18	Predicate	37
2.19	Priority	37
2.20	Property List	37

Contents

2.21	Quantity	37
2.22	Quantity_Range	38
2.23	Record	38
2.24	Reference	38
2.25	Retention	38
2.26	Safety_Stock_Units	39
2.27	String	39
2.28	Symbol	39
2.29	Time	39
2.30	Type	40
2.31	Typed_Value	40
2.32	Void	40
3	Basic Functions	41
3.1	Operators	41
3.2	List Functions	44
3.2.1	count (List Void)	45
3.2.2	filter (List Void, Expression, Integer)	45
3.2.3	for_each (List Void, Expression)	46
3.2.4	sort (List Void, Expression)	46
3.2.5	sort_stable (List Void, Expression)	47
3.2.6	sublist (List Void, Integer, Integer)	48
3.2.7	unique (List Void)	48
3.2.8	contains (List Void, Void)	49
3.2.9	found (List Void, Void)	49
3.2.10	clement (List Void, Integer)	50
3.2.11	find (List Void, Void)	50
3.2.12	find_or_nonexistent (List Void, Void)	50
3.2.13	find_or_create (List Void, Void)	51
3.2.14	list_index (List Void, Void)	51
3.2.15	first (List Void)	52
3.2.16	last (List Void)	52
3.2.17	list ()	52
3.2.18	integers (Integer, Integer)	53
3.3	Multi_Keyed_List Functions	53
3.3.1	key (Symbol, Symbol, Logical, Logical)	54
3.3.2	multi_key (Symbol, Void)	54
3.3.3	multi_keyed_list (List Void, List Void, Expression)	55
3.3.4	multi_keyed_clement (List Void, Integer, Symbol, Symbol)	55
3.3.5	key_names (List Void)	56
3.3.6	key_values (List Void, Symbol)	56
3.3.7	multi_key_bucketize (List Void, List Void, List Void, Range, Expression)	56
3.3.8	multi_key_bucketize_clement (List Void, Integer, Date, Symbol, Symbol)	57
3.4	Singly_Keyed_List Functions	57
3.4.1	bucketize (List Void, List Void, Date, Range, Expression)	58
3.4.2	bucket_list (List Void, Date, Symbol)	59
3.4.3	bucket_list_by_date (List Void, Date)	60
3.4.4	bucket_list_by_key (List Void, Symbol)	61
3.4.5	bucket_symbols (List Void)	62
3.4.6	keyed_list (List Void, Expression)	62
3.4.7	keyed_clement (Symbol)	63

Contents

3.4.8	keys (List Void)	63
3.5	Recurse_List Functions	64
3.5.1	recurse (List Void, Expression)	64
3.5.2	recurse_and_urn (List Void, Expression, Expression)	65
3.5.3	current_depth (Cell)	66
3.6	Text_String Functions	66
3.6.1	string (Void)	66
3.6.2	string (Void, Symbol)	67
3.6.3	index (String, String)	67
3.6.4	index (String, String, Integer)	68
3.6.5	left (String)	68
3.6.6	left (String, Integer)	68
3.6.7	left (String, String)	69
3.6.8	lower (String)	69
3.6.9	mid (String, Integer)	69
3.6.10	mid (String, Integer, Integer)	70
3.6.11	mid (String, String, Integer)	70
3.6.12	proper (String)	70
3.6.13	justify (String, Integer, String)	71
3.6.14	replace (String, Integer, Integer, String)	71
3.6.15	right (String, String)	72
3.6.16	right (String)	72
3.6.17	right (String, Integer)	72
3.6.18	upper (String)	73
3.6.19	urn (String)	73
3.6.20	urn_left (String)	73
3.6.21	urn_right (String)	73
3.6.22	code (String)	74
3.6.23	code (String, Integer)	74
3.6.24	len (String)	74
3.6.25	match_count (String, String)	75
3.6.26	concat (String, String)	75
3.6.27	whitespace (String, String)	76
3.6.28	logical (String)	76
3.6.29	logical (String, Symbol)	76
3.6.30	ischar (Integer)	77
3.6.31	concat (Symbol, Symbol)	77
3.6.32	symbol (String)	77
3.6.33	symbol (String, Symbol)	78
3.7	Numeric Functions	78
3.7.1	abs (Integer)	78
3.7.2	abs (Number)	78
3.7.3	abs (Quantity)	79
3.7.4	abs (Time)	79
3.7.5	finite (Number)	79
3.7.6	concat (Quantity, Quantity)	80
3.7.7	cos (Number)	80
3.7.8	csp (Number)	80
3.7.9	integer (Number)	81
3.7.10	integer (Percentage)	81

Contents

3.7.11	integer (String).....	81
3.7.12	integer (String, Symbol).....	82
3.7.13	in (Number).....	82
3.7.14	log (Number).....	82
3.7.15	log (Number, Number).....	82
3.7.16	log10 (Number).....	83
3.7.17	measure (Quantity).....	83
3.7.18	measure (String).....	83
3.7.19	measure (String, Symbol).....	84
3.7.20	money (String).....	84
3.7.21	money (String, Symbol).....	85
3.7.22	number (Integer).....	85
3.7.23	number (Percentage).....	85
3.7.24	number (Quantity).....	85
3.7.25	number (String).....	86
3.7.26	number (String, Symbol).....	86
3.7.27	percentage (Integer).....	86
3.7.28	percentage (Number).....	87
3.7.29	percentage (Quantity).....	87
3.7.30	percentage (String).....	87
3.7.31	percentage (String, Symbol).....	88
3.7.32	power (Number, Number).....	88
3.7.33	quantity (String).....	88
3.7.34	quantity (Integer).....	88
3.7.35	quantity (Number).....	89
3.7.36	quantity (String).....	89
3.7.37	quantity (String, Symbol).....	90
3.7.38	quantity (Time).....	90
3.7.39	round (Number).....	90
3.7.40	round (Number, Number).....	90
3.7.41	round_down (Number).....	91
3.7.42	round_down (Number, Number).....	91
3.7.43	round_in (Number).....	91
3.7.44	round_in (Number, Number).....	92
3.7.45	round_out (Number).....	92
3.7.46	round_out (Number, Number).....	92
3.7.47	round_up (Number).....	93
3.7.48	round_up (Number, Number).....	93
3.7.49	sin (Number).....	94
3.7.50	sqrt (Number).....	94
3.7.51	tan (Number).....	94
3.7.52	time (Quantity).....	94
3.7.53	time (String).....	95
3.7.54	time (String, Symbol).....	95
3.7.55	units (Quantity).....	96
3.8	Random Functions.....	96
3.8.1	rand_integer (Integer, Integer).....	96
3.8.2	rand ().....	96
3.8.3	rand (Number, Number).....	97
3.8.4	rand_seed (Number).....	97

Contents

3.9	Quantity, Range Functions.....	97
3.9.1	intersect (Quantity, Range, Quantity, Range).....	97
3.9.2	within (Quantity, Quantity, Range).....	98
3.9.3	within (Quantity, Range, Quantity, Range).....	99
3.9.4	interval (Quantity, Range).....	99
3.9.5	limit (Quantity, Quantity, Range).....	100
3.9.6	limit (Quantity, Range, Quantity, Range).....	100
3.9.7	max (Quantity, Range).....	101
3.9.8	min (Quantity, Range).....	101
3.9.9	encluse (Quantity, Quantity, Range).....	102
3.9.10	encluse (Quantity, Range, Quantity, Range).....	102
3.9.11	quantity_interval (Quantity, Quantity).....	103
3.9.12	quantity_range (Quantity, Quantity).....	103
3.9.13	quantity_range (String).....	104
3.9.14	quantity_range (String, Symbol).....	104
3.9.15	sel_max (Quantity, Range, Quantity).....	105
3.9.16	sel_min (Quantity, Range, Quantity).....	105
3.9.17	set_interval (Quantity, Range, Quantity).....	106
3.10	Numeric List Functions.....	106
3.10.1	average (List(Integer)).....	106
3.10.2	average (List(Integer)).....	107
3.10.3	average (List(Percentage)).....	107
3.10.4	average (List(Quantity)).....	107
3.10.5	average (List(Time)).....	108
3.10.6	max (List(Integer)).....	108
3.10.7	max (List(Integer)).....	108
3.10.8	max (List(Percentage)).....	109
3.10.9	max (List(Quantity)).....	109
3.10.10	max (List(Time)).....	109
3.10.11	min (List(Integer)).....	109
3.10.12	min (List(Integer)).....	110
3.10.13	min (List(Percentage)).....	110
3.10.14	min (List(Quantity)).....	110
3.10.15	min (List(Time)).....	111
3.10.16	product (List(Integer)).....	111
3.10.17	product (List(Integer)).....	111
3.10.18	product (List(Percentage)).....	112
3.10.19	product (List(Quantity)).....	112
3.10.20	sidev (List(Integer)).....	112
3.10.21	sidev (List(Integer)).....	113
3.10.22	sidev (List(Percentage)).....	113
3.10.23	sidev (List(Quantity)).....	113
3.10.24	sidev (List(Time)).....	114
3.10.25	sum (List(Integer)).....	114
3.10.26	sum (List(Integer)).....	114
3.10.27	sum (List(Percentage)).....	115
3.10.28	sum (List(Quantity)).....	115
3.10.29	sum (List(Time)).....	115
3.11	Date Functions.....	115
3.11.1	max (List(Date)).....	116

Contents

3.11.2	min (List Date).....	116
3.11.3	date (String).....	116
3.11.4	date (String, Symbol).....	117
3.11.5	now ().....	117
3.11.6	start_of_day (Date).....	118
3.11.7	start_of_day (Date, Integer).....	118
3.11.8	start_of_month (Date).....	118
3.11.9	start_of_month (Date, Integer).....	119
3.11.10	start_of_week (Date).....	119
3.11.11	start_of_week (Date, Integer).....	119
3.11.12	start_of_year (Date).....	120
3.11.13	start_of_year (Date, Integer).....	120
3.11.14	horizon_date (String).....	120
3.11.15	finite (Date).....	122
3.11.16	day_of_month (Date).....	122
3.11.17	day_of_week (Date).....	122
3.11.18	day_of_year (Date).....	122
3.11.19	hour (Date).....	123
3.11.20	minute (Date).....	123
3.11.21	month (Date).....	123
3.11.22	number_of_weeks (Date, Date).....	124
3.11.23	second (Date).....	124
3.11.24	week_of_year (Date).....	124
3.11.25	year (Date).....	124
3.11.26	within_daylight_savings_time (Date).....	125
3.11.27	within_leap_year (Date).....	125
3.11.28	closing_day (Date).....	125
3.11.29	date (Horizon, Date, Date).....	126
3.11.30	restriction (String).....	126
3.11.31	restriction (String, Date).....	126
3.11.32	restriction (String, Date, Range).....	126
3.11.33	satisfies (Restriction, Date, Range).....	127
3.12	Date, Range Functions.....	127
3.12.1	date_range (Date, Date).....	127
3.12.2	date_range (Date, Time).....	127
3.12.3	date_range (String).....	128
3.12.4	date_range (String, Symbol).....	128
3.12.5	enclose (Date, Date, Range).....	128
3.12.6	enclose (Date, Range, Date, Range).....	129
3.12.7	end (Date, Range).....	129
3.12.8	intersect (Date, Range, Date, Range).....	129
3.12.9	limit (Date, Date, Range).....	131
3.12.10	limit (Date, Range, Date, Range).....	131
3.12.11	rel_end (Date, Range, Date).....	132
3.12.12	rel_start (Date, Range, Date).....	132
3.12.13	set_time (Date, Range, Time).....	133
3.12.14	start (Date, Range).....	133
3.12.15	time (Date, Range).....	134
3.12.16	within (Date, Date, Range).....	134
3.12.17	within (Date, Range, Date, Range).....	135

Contents

3.13	Date, Range, List Functions.....	135
3.13.1	days (Date, Range).....	135
3.13.2	days (Date, Range, Time).....	136
3.13.3	months (Date, Range).....	137
3.13.4	quarters (Date, Range).....	138
3.13.5	shifts (Date, Range).....	138
3.13.6	shifts (Date, Range, Time).....	139
3.13.7	shifts (Date, Range, Time, Time).....	140
3.13.8	weeks (Date, Range).....	141
3.13.9	weeks (Date, Range, Integer).....	142
3.14	Logical Functions.....	143
3.14.1	and (Logical, Logical).....	143
3.14.2	not (Logical).....	143
3.14.3	or (Logical, Logical).....	144
3.14.4	xor (Logical, Logical).....	144
3.15	Miscellaneous Functions.....	145
3.15.1	id (Void).....	145
3.15.2	current_index (Cell).....	145
3.15.3	system (String).....	146
3.15.4	values (Type).....	146
3.15.5	data (Symbol).....	146
3.15.6	ctatus (Void).....	146
3.15.7	exists_without_crosses (Void).....	147
3.15.8	export (string, void).....	147
3.15.9	import (string, void).....	147
3.15.10	engine_name ().....	148
3.15.11	get_color (String).....	148
3.15.12	get_date_string (Integer).....	149
3.15.13	getenv (String).....	149
3.15.14	uname (String).....	149
3.15.15	model_file ().....	149
3.15.16	option (String).....	149
3.15.17	setenv (String, String).....	150
3.15.18	typed_value (void).....	150
3.15.19	owner (Typed, Value).....	151
3.15.20	first_value (Typed, Value).....	151
3.15.21	delic (Void).....	151
3.15.22	make_type (Void, Type).....	151
3.15.23	nonexistent ().....	152
3.15.24	set_option (String, String).....	152
3.15.25	type (Typed, Value).....	152
3.15.26	type (String).....	152
3.15.27	model_type (void).....	153
3.15.28	value (Typed, Value, Type).....	153
3.15.29	trace_time ().....	153
3.15.30	memory_size ().....	154
3.15.31	process_size ().....	154
3.15.32	with_connection (Connection, void).....	155
3.15.33	functions (Symbol).....	155
3.15.34	echo (Logical).....	156

Contents

3.15.35	echo (String)	156
3.15.36	with_echo_file (String, void)	156
3.16	File Functions	156
3.16.1	is_directory (String)	156
3.16.2	is_file (String)	157
3.16.3	is_writable (String)	157
3.17	Evaluation Functions	157
3.17.1	background (Expression)	157
3.17.2	do ()	158
3.17.3	call (Symbol)	158
3.17.4	do_file_echo (String)	158
3.17.5	do_file (String)	159
3.17.6	do_file_fast (String)	159
3.17.7	engine (Expression)	159
3.17.8	evaluate (Expression)	160
3.17.9	if (Logical, Void, Void)	160
3.17.10	while (Logical, Void)	160
3.17.11	reference (Variable)	160
3.18	RhythmLink Functions	161
3.18.1	rl_field (Data_Table, String)	161
3.18.2	rhythmlink_field (Symbol, String)	162
3.18.3	rl_record_field (Record, String)	162
3.19	Debug Functions	162
3.19.1	stop (Symbol, Symbol)	162
3.19.2	break (Symbol)	163
3.19.3	break (Expression, Symbol, Expression)	163
3.19.4	breakpoint ()	163
3.19.5	breakpoint (Symbol)	164
3.19.6	disable ()	164
3.19.7	disable (Symbol)	164
3.19.8	enable ()	164
3.19.9	enable (Symbol)	164
3.19.10	next ()	164
3.19.11	step ()	164
3.19.12	cont ()	164
3.19.13	next (Integer)	164
3.19.14	stop (Integer)	165
3.19.15	cont (Integer)	165
3.19.16	watch (Expression, Symbol, Expression)	165
3.19.17	unwatch (Symbol)	165
3.19.18	where ()	165
3.19.19	wherists ()	165
3.19.20	println_variables ()	165
3.19.21	println_report_variables ()	166
3.19.22	inspect (void)	166
3.19.23	inspect (Integer)	166
3.19.24	inspect (Integer, Integer)	166
3.19.25	inspectV (Integer)	166
3.19.26	inspectV (Integer, Integer)	167
3.19.27	inspectV (Integer, Integer)	167

Contents

3.20	Program Functions	167
3.20.1	user ()	167
3.20.2	quit (String)	167
3.20.3	shutdown (String)	168
3.21	Engine Queue Functions	168
3.21.1	wait ()	168
4	Summary Section	169
5	Models	169
5.1	Supply Chain Model	228
5.1.1	Site Model	232
5.1.1.1	Standard Extensions of model Site	235
5.1.1.1.1	role extensions of model Site	242
5.1.1.1.1.1	LINK Extension	242
5.1.1.1.1.2	SUPPLIER Extension	244
5.1.1.1.1.3	CUSTOMER Extension	245
5.1.1.2	role submodels of model Site	245
5.1.1.3	Location Model	246
5.1.1.4	Item Model	246
5.1.1.4.1	spec submodels of model Item	249
5.1.1.5	Buffer Model	254
5.1.1.5.1	Buffer_Problem_Detector Model	254
5.1.1.5.2	flow_policy submodels of model Buffer	260
5.1.1.5.3	Flow_Criterion Model	262
5.1.1.6	Resource Model	262
5.1.1.6.1	Resource_Skill Model	263
5.1.1.6.2	efficiency submodels of model Resource	270
5.1.1.6.3	size submodels of model Resource	271
5.1.1.6.4	Size_Dimension Model	272
5.1.1.6.5	load_policy submodels of model Resource	273
5.1.1.6.6	Resource_Setup_Order Model	273
5.1.1.6.7	Resource_Blocks Model	273
5.1.1.7	Skill Model	273
5.1.1.7.1	Skill_Resource Model	275
5.1.1.8	Operation Model	276
5.1.1.8.1	Load Model	288
5.1.1.8.1.1	Load_Size Model	291
5.1.1.8.2	Flow Model	292
5.1.1.8.2.1	usage_policy submodels of model Flow	294
5.1.1.8.3	Operation_Problem_Detector Model	295
5.1.1.8.4	process submodels of model Operation	297
5.1.1.8.5	Alternate_Operation Model	297
5.1.1.8.6	Effective_Calendar_Operation Model	298
5.1.1.8.7	Routing_Operation Model	299
5.1.1.9	Configuration Model	300
5.1.1.10	spec submodels of model Configuration	302
5.1.1.10.1	Item_Group Model	302
5.1.1.10.2	Sub_Item_Group Model	305
5.1.2	Sub_Item Model	306
5.1.2.1	Seller Model	307
5.1.2.1	Site_Group Model	313

Contents

5.1.2.1.1	Explicit_Site_Model	315
5.1.2.2	Product_Rent_Model	316
5.1.2.2.1	Product_Supplier_Model	320
5.1.2.2.1.1	Product_Item_Model	321
5.1.2.3	Product_Model	322
5.1.2.3.2	Generic_Product_Model	334
5.1.2.3.2	Alternate_Product_Model	335
5.1.2.3.3	allocation_policy_submodels_of_model_Product	336
5.1.2.3.4	Product_Allocation_Model	336
5.1.2.4	Product_Group_Model	337
5.1.2.4.1	Sub_Product_Group_Model	341
5.1.2.4.2	Sub_Product_Model	343
5.1.3	Plan_Model	345
5.1.3.1	Site_Plan_Model	350
5.1.3.1.1	Standard Extensions of model Site_Plan	352
5.1.3.1.1.1	role extensions of model Site_Plan	352
5.1.3.1.1.2	LINK_Extension	352
5.1.3.1.1.2	SUPPLIER_Extension	355
5.1.3.1.1.3	CUSTOMER_Extension	355
5.1.3.1.2	role_submodels_of_model_Site_Plan	356
5.1.3.1.3	Buffer_Plan_Model	356
5.1.3.1.3.1	Lot_Model	361
5.1.3.1.3.2	flow_policy_submodels_of_model_Buffer_Plan	363
5.1.3.1.3.3	Sorted_Bucket_Model	363
5.1.3.1.4	Resource_Plan_Model	364
5.1.3.1.4.1	load_policy_submodels_of_model_Resource_Plan	373
5.1.3.1.4.2	Consolidation_Model	373
5.1.3.1.5	Operation_Plan_Model	373
5.1.3.1.5.1	Load_Plan_Model	381
5.1.3.1.5.2	Flow_Plan_Model	383
5.1.3.1.5.2.1	Lot_Flow_Model	385
5.1.3.1.6	Operation_State_Model	386
5.1.3.1.7	Request_Model	388
5.1.3.1.7.1	Delivery_Request_Model	397
5.1.3.1.7.1.1	Item_Request_Model	405
5.1.3.1.8	Promise_Model	411
5.1.3.1.8.1	Delivery_Promise_Model	416
5.1.3.1.8.1.1	Delivery_Available_To_Promise_Model	424
5.1.3.1.8.1.2	Item_Promise_Model	426
5.1.3.1.8.1.2.1	Item_Promise_Model	433
5.1.3.1.8.1.2.1.1	Product_Available_To_Promise_Model	435
5.1.3.1.9	Acceptance_Model	437
5.1.3.1.9.1	Delivery_Acceptance_Model	440
5.1.3.1.9.1.1	Item_Acceptance_Model	444
5.1.3.2	Seller_Plan_Model	449
5.1.3.2.1	Forecast_Model	453
5.1.3.2.1.1	Standard Extensions of model Forecast	459
5.1.3.2.1.1.1	level_extensions_of_model_Forecast	459
5.1.3.2.1.1.1.1	INDIVIDUAL_Extension	459
5.1.3.2.1.1.1.2	GROUP_Extension	459

Contents

5.1.3.2.1.2	Forecast_Entry_Model	461
5.1.3.2.1.3	level_submodels_of_model_Forecast	478
5.1.3.2.1.4	ATP_Entry_Model	478
5.1.3.3	Problem_Model	480
5.1.3.4	Active_Strategy_Model	483
5.1.3.4.1	Active_Problem_Model	488
5.1.3.4.2	Active_Goal_Model	490
5.2	Problem_Category_Model	492
5.3	Horizon_Model	494
5.3.1	bucket_spec_submodels_of_model_Horizon	496
5.3.2	Horizon_Bucket_Start_Model	496
5.4	Strategy_Model	497
5.4.1	Problem_Sci_Model	502
5.4.2	Strategy_Change_Model	505
5.4.3	Strategy_Lock_Model	507
5.4.4	Strategy_Goal_Model	508
5.4.5	execution_submodels_of_model_Strategy	509
5.4.6	Ordered_Sub_Strategy_Model	509
5.4.7	Ranked_Sub_Strategy_Model	509
5.5	Unit_Model	511
5.5.1	Unit_Quantity_Model	513
5.6	Profile_Number_Model	514
5.7	Profile_Percentage_Model	515
5.8	Profile_Quantity_Model	516
5.9	Box_Model	517
5.10	Calendar_Model	519
5.10.1	Calendar_Entry_Model	522
5.10.2	Sub_Calendar_Model	526
5.11	Calendar_Plan_Model	528
6	Control_Model	530
7	Connection_Model	531
7.1	User_Model	532
7.1.1	Report_Directory_Model	536
7.1.2	Report_Model	538
7.1.3	Layout_Model	539
7.1.4	Worksheet_Model	540
7.1.5	Style_Model	541
7.1.6	Format_Model	542
7.1.7	Domain_Model	544
7.2	Model_Type_Model	545
7.2.1	Field_Model	547
7.2.2	Extension_Selector_Model	550
7.3	Field_Error_Model	551
8	Function_Model	553
9	Breakpoint_Model	554
10	Extensions	555
10.1	Site_Extensions	568
10.1.1	role_extensions_of_model_Site	568
10.1.1.1	LINK_Extension	568
10.1.1.2	SUPPLIER_Extension	570

Contents

10.1.1.3	CUSTOMER Extension.....	570
10.2	Seller Extensions.....	572
10.2.1	request_naming extensions of model Seller.....	572
10.2.1.1	NONE Extension.....	572
10.3	Sic Group Extensions.....	573
10.3.1	spec extensions of model Sic_Group.....	573
10.4	Product_Root Extensions.....	574
10.4.1	forecast_policy extensions of model Product_Root.....	574
10.4.1.1	SIMPLE_FIXED_QUANTITY Extension.....	574
10.4.1.2	SINGLE_REQUEST Extension.....	575
10.4.1.3	SIMPLE_FIXED_TIME Extension.....	575
10.4.1.4	WEEKLY Extension.....	577
10.4.1.5	DUAL_REQUEST Extension.....	578
10.5	Product Extensions.....	580
10.5.1	forecast_policy extensions of model Product.....	580
10.5.1.1	SIMPLE_FIXED_QUANTITY Extension.....	580
10.5.1.2	SINGLE_REQUEST Extension.....	580
10.5.1.3	SIMPLE_FIXED_TIME Extension.....	580
10.5.1.4	WEEKLY Extension.....	580
10.5.1.5	DUAL_REQUEST Extension.....	580
10.5.2	expiration_policy extensions of model Product.....	580
10.5.2.1	AT_END Extension.....	580
10.5.3	allocation_policy extensions of model Product.....	580
10.5.3.1	FCTS Extension.....	580
10.5.3.2	PER_ALLOCATED Extension.....	581
10.5.3.3	PER_COMMITTED Extension.....	581
10.5.3.4	MEMBER_RANK Extension.....	581
10.5.3.5	FIXED_SPLIT Extension.....	583
10.5.4	availability_policy extensions of model Product.....	584
10.5.4.1	SLIDING Extension.....	584
10.5.4.2	HORIZON Extension.....	584
10.5.5	price_policy extensions of model Product.....	585
10.5.5.1	NONE Extension.....	585
10.5.5.2	FIXED Extension.....	585
10.6	Operation Extensions.....	586
10.6.1	process extensions of model Operation.....	586
10.6.1.1	ALTERNATES_PRIMARY Extension.....	586
10.6.1.2	ALTERNATES_PROPORTIONAL Extension.....	586
10.6.1.3	EFFECTIVE_CALENDAR Extension.....	587
10.6.1.4	DELAY_ONLY_FIXED Extension.....	588
10.6.1.5	DELAY_ONLY_BASIC Extension.....	588
10.6.1.6	BASIC_CALENDARS Extension.....	588
10.6.1.7	FIXED_TIME Extension.....	589
10.6.1.8	TIME_MULTIPLE Extension.....	590
10.6.1.9	BASIC Extension.....	590
10.6.1.10	BASIC_DELAYED Extension.....	590
10.6.1.11	REQUEST_FIXED Extension.....	591
10.6.1.12	REQUEST_FIXED_WITH_ANALYSIS Extension.....	593
10.6.1.13	ROUTING Extension.....	594
10.7	Load Extensions.....	596

Contents

10.7.1	usage_policy extensions of model Load.....	596
10.7.1.1	RESOURCE Extension.....	596
10.7.1.2	ONE Extension.....	596
10.8	Load_Size Extensions.....	597
10.8.1	load_size_usage extensions of model Load_Size.....	597
10.8.1.1	FIXED Extension.....	597
10.8.1.2	LINEAR Extension.....	597
10.9	Flow Extensions.....	598
10.9.1	usage_policy extensions of model Flow.....	598
10.9.1.1	CONSUME_FIXED Extension.....	598
10.9.1.3	CONSUME_PER Extension.....	598
10.9.1.4	PRODUCE_FIXED Extension.....	598
10.9.1.5	PRODUCE_PER Extension.....	599
10.9.1.6	PRODUCE_YIELD Extension.....	599
10.10	PRODUCE_YIELD_CALENDAR Extension.....	600
10.10.1	Operation_Problem_Decider Extensions.....	602
10.11	detector extensions of model Operation_Problem_Decider.....	602
10.11.1	Operation_Plan Extensions.....	603
10.11.1.1	move extensions of model Operation_Plan.....	603
10.11.1.2	PRODUCE Extension.....	603
10.11.1.3	CONSUME Extension.....	603
10.11.1.4	DELIVER Extension.....	604
10.11.2	RECEIVE Extension.....	605
10.11.2.1	process extensions of model Operation_Plan.....	605
10.11.2.2	ALTERNATES_PRIMARY Extension.....	605
10.11.2.3	ALTERNATES_PROPORTIONAL Extension.....	606
10.11.2.4	EFFECTIVE_CALENDAR Extension.....	606
10.11.2.5	DELAY_ONLY_FIXED Extension.....	606
10.11.2.6	DELAY_ONLY_BASIC Extension.....	607
10.11.2.7	BASIC_CALENDARS Extension.....	607
10.11.2.8	FIXED_TIME Extension.....	607
10.11.2.9	TIME_MULTIPLE Extension.....	607
10.11.2.10	BASIC_EXTENSION Extension.....	607
10.11.2.11	REQUEST_FIXED Extension.....	607
10.11.2.12	REQUEST_FIXED_WITH_ANALYSIS Extension.....	608
10.12	ROUTING Extension.....	608
10.12.1	Resource Extensions.....	610
10.12.1.1	efficiency extensions of model Resource.....	610
10.12.1.2	FIXED Extension.....	610
10.12.2	CALENDAR Extension.....	610
10.12.2.1	variability extensions of model Resource.....	610
10.12.2.2	ZERO Extension.....	610
10.12.3	FIXED Extension.....	610
10.12.3.1	maintenance extensions of model Resource.....	611
10.12.4	ZERO Extension.....	611
10.12.4.1	size extensions of model Resource.....	611
10.12.4.2	UNLIMITED Extension.....	611
10.12.4.3	FIXED_COUNT Extension.....	612
10.12.4.3	FIXED_QUANTITY Extension.....	612

Contents

10.12.4	CALENDAR COUNT Extension.....	612
10.12.4.5	MULTI_DIMENSION Extension.....	613
10.12.5	load_policy extensions of model Resource.....	614
10.12.5.1	SIMPLE_CONSOLIDATION Extension.....	614
10.12.5.2	INFINITE_USE Extension.....	614
10.12.5.3	EXCLUSIVE_USE Extension.....	614
10.12.5.4	SHARED_USE Extension.....	614
10.13	Resource_Skill Extensions.....	617
10.13.1	efficiency extensions of model Resource_Skill.....	617
10.13.1.1	FIXED Extension.....	617
10.13.1.2	CALENDAR Extension.....	617
10.14	Resource_Problem_Detector Extensions.....	618
10.14.1	detector extensions of model Resource_Problem_Detector.....	618
10.15	Resource_Plan Extensions.....	619
10.15.1	efficiency extensions of model Resource_Plan.....	619
10.15.2	load_policy extensions of model Resource_Plan.....	619
10.15.2.1	SIMPLE_CONSOLIDATION Extension.....	619
10.15.2.2	INFINITE_USE Extension.....	619
10.15.2.3	EXCLUSIVE_USE Extension.....	619
10.15.2.4	SHARED_USE Extension.....	619
10.15.3	size extensions of model Resource_Plan.....	620
10.15.4	maintenance extensions of model Resource_Plan.....	620
10.16	Buffer Extensions.....	621
10.16.1	flow_policy extensions of model Buffer.....	621
10.16.1.1	BUCKETED_NESTED_SORT Extension.....	621
10.16.1.2	PRODUCING_FLOW_CALENDAR Extension.....	621
10.16.1.3	PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK Extension.....	623
10.16.1.4	SUPPLY_CALENDAR Extension.....	628
10.16.1.5	ON_HAND_CALENDAR Extension.....	637
10.16.1.6	ON_HAND_CALENDAR_FILTER_AND_RANK Extension.....	638
10.16.1.7	FLOW_LIMIT_CALENDAR Extension.....	642
10.16.1.8	FLOW_LIMIT_CALENDAR_FILTER_AND_RANK Extension.....	643
10.16.1.9	INFINITE Extension.....	647
10.16.1.10	SUPPLIER Extension.....	647
10.16.1.11	BASIC Extension.....	648
10.16.1.12	BASIC_FILTER_AND_RANK Extension.....	648
10.16.1.13	FIXED_QUANTITY Extension.....	654
10.16.1.14	MULTIPLE Extension.....	656
10.16.1.15	MULTIPLE_FILTER_AND_RANK Extension.....	657
10.16.1.16	FIXED_QUANTITY_FENCED Extension.....	663
10.16.1.17	FIXED_TIME Extension.....	665
10.16.1.18	LF_SIMPLE Extension.....	666
10.16.1.19	LF_BOUNDED Extension.....	667
10.16.1.20	MLFL_BOUNDED Extension.....	669
10.16.2	stocking_policy extensions of model Buffer.....	671
10.16.2.1	MANUAL Extension.....	671
10.16.2.2	CALENDAR Extension.....	671
10.17	Buffer_Problem_Detector Extensions.....	676
10.17.1	detector extensions of model Buffer_Problem_Detector.....	676
10.18	Buffer_Plan Extensions.....	677

Contents

10.18.1	flow_policy extensions of model Buffer_Plan.....	677
10.18.1.1	BUCKETED_NESTED_SORT Extension.....	677
10.18.1.2	PRODUCING_FLOW_CALENDAR Extension.....	677
10.18.1.3	PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK Extension.....	681
10.18.1.4	ON_HAND_CALENDAR Extension.....	685
10.18.1.5	ON_HAND_CALENDAR_FILTER_AND_RANK Extension.....	685
10.18.1.6	FLOW_LIMIT_CALENDAR Extension.....	686
10.18.1.7	FLOW_LIMIT_CALENDAR_FILTER_AND_RANK Extension.....	686
10.18.1.8	BASIC Extension.....	686
10.18.1.9	BASIC_FILTER_AND_RANK Extension.....	690
10.18.1.10	FIXED_QUANTITY Extension.....	694
10.18.1.11	MULTIPLE Extension.....	697
10.18.1.12	MULTIPLE_FILTER_AND_RANK Extension.....	701
10.18.1.13	FIXED_QUANTITY_FENCED Extension.....	705
10.18.1.14	FIXED_TIME Extension.....	709
10.19	Forecast Extensions.....	713
10.19.1	level extensions of model Forecast.....	713
10.19.1.1	INDIVIDUAL Extension.....	713
10.19.1.2	GROUP Extension.....	713
10.20	Forecast_Entry Extensions.....	715
10.20.1	forecast_policy extensions of model Forecast_Entry.....	715
10.20.2	allocation_policy extensions of model Forecast_Entry.....	715
10.20.2.1	MEMBER_RANK Extension.....	715
10.21	Sic_Plan Extensions.....	716
10.21.1	role extensions of model Sic_Plan.....	716
10.21.1.1	LINK Extension.....	716
10.21.1.2	SUPPLIER Extension.....	719
10.21.1.3	CUSTOMER Extension.....	719
10.22	Problem Extensions.....	720
10.22.1	category extensions of model Problem.....	720
10.22.1.1	REQUEST_NOT_PLANNED Extension.....	720
10.22.1.2	REQUEST_PLANNED_LATE Extension.....	721
10.22.1.3	REQUEST_PLANNED_EARLY Extension.....	723
10.22.1.4	REQUEST_PLANNED_SHORT Extension.....	725
10.22.1.5	REQUEST_PLANNED_EXCESS Extension.....	726
10.22.1.6	PROMISE_NOT_PLANNED Extension.....	728
10.22.1.7	PROMISE_PLANNED_LATE Extension.....	730
10.22.1.8	PROMISE_PLANNED_EARLY Extension.....	731
10.22.1.9	PROMISE_PLANNED_SHORT Extension.....	733
10.22.1.10	PROMISE_PLANNED_EXCESS Extension.....	734
10.22.1.11	ACCEPTANCE_NOT_PLANNED Extension.....	736
10.22.1.12	ACCEPTANCE_PLANNED_LATE Extension.....	738
10.22.1.13	ACCEPTANCE_PLANNED_EARLY Extension.....	739
10.22.1.14	ACCEPTANCE_PLANNED_SHORT Extension.....	741
10.22.1.15	ACCEPTANCE_PLANNED_EXCESS Extension.....	743
10.22.1.16	PLANNED_BEFORE_CURRENT Extension.....	744
10.22.1.17	UNRELEASED Extension.....	745
10.22.1.18	NEEDS_RELEASE Extension.....	745
10.22.1.19	INCONSISTENT_OPPLAN Extension.....	745
10.22.1.20	REQUEST_PROMISED_LATE Extension.....	746

Contents

10.22.1.21	REQUEST_PROMISED_EARLY	Extension	747
10.22.1.22	REQUEST_PROMISED_SHORT	Extension	748
10.22.1.23	REQUEST_PROMISED_EXCESS	Extension	750
10.22.1.24	ITEM_PROMISE_OVERPRICED	Extension	751
10.22.1.25	DELIVERY_PROMISE_OVERPRICED	Extension	752
10.22.1.26	PROMISE_NOT_OFFERED	Extension	753
10.22.1.27	PROMISE_NOT_ACCEPTED	Extension	754
10.22.1.28	ACCEPTANCE_INCONSISTENT	Extension	755
10.22.1.29	REQUEST_QUEUED	Extension	755
10.22.1.30	DELIVERY_REQUEST_NOT_COORDINATED	Extension	756
10.22.1.31	DELIVERY_PROMISE_NOT_COORDINATED	Extension	757
10.22.1.32	DELIVERY_ACCEPTANCE_NOT_COORDINATED	Extension	758
10.22.1.33	NEGATIVE_ATP	Extension	759
10.22.1.34	NEGATIVE_PLANNED_ATP	Extension	760
10.22.1.35	OVER_COMMITTED	Extension	760
10.22.1.36	OVER_CONSUMED	Extension	761
10.22.1.37	UNALLOCATED_FORECAST	Extension	761
10.22.1.38	SUPPLY_PLANNED_LATE	Extension	762
10.22.1.39	SUPPLY_PLANNED_EARLY	Extension	763
10.22.1.40	SUPPLY_PLANNED_SHORT	Extension	765
10.22.1.41	SUPPLY_PLANNED_EXCESS	Extension	767
10.22.1.42	SUPPLY_PROMISED_LATE	Extension	769
10.22.1.43	SUPPLY_PROMISED_EARLY	Extension	770
10.22.1.44	SUPPLY_PROMISED_SHORT	Extension	772
10.22.1.45	SUPPLY_PROMISED_EXCESS	Extension	773
10.22.1.46	UNIDENTIFIED_OP_STATE	Extension	775
10.22.1.47	UNCOORDINATED	Extension	775
10.22.1.48	UNCOORDINATED	Extension	776
10.22.1.49	CONSOLIDATION_OVERSIZE	Extension	776
10.22.1.50	CONSOLIDATION_UNDERSIZE	Extension	777
10.22.1.51	OVERLOAD	Extension	778
10.22.1.52	OVERSIZE	Extension	778
10.22.1.53	BUCKET_OVERSIZE	Extension	779
10.22.1.54	UNDERLOAD	Extension	779
10.22.1.55	NEGATIVE_ON_HAND	Extension	780
10.22.1.56	OVER_FLOW_LIMIT	Extension	781
10.22.1.57	NEGATIVE_ON_HAND_AT_END	Extension	782
10.22.1.58	LOT_OVER_CONSUMED	Extension	782
10.22.1.59	LOT_NOT_CONSUMED	Extension	783
10.22.1.60	LOT_NOT_PRODUCED	Extension	783
10.22.1.61	LOT_OVER_PRODUCED	Extension	783
10.22.1.62	LOW_ON_HAND	Extension	784
10.22.1.63	EXCESS_ON_HAND	Extension	784
10.22.1.64	EXCESS_ON_HAND_AT_END	Extension	785
10.23	Active_Strategy Extensions		786
10.23.1	Termination extensions of model Active_Strategy		786
10.23.1.1	NO_PROBLEMS	Extension	786
10.23.1.2	TARGET_ACHIEVED	Extension	786
10.23.1.3	RESOLVE_COUNT_EXCEEDED	Extension	786
10.23.1.4	MANUAL	Extension	786

Contents

10.23.2	execution extensions of model Active_Strategy	786	
10.23.2.1	SEQUENCE_RUN_ONCE	Extension	786
10.23.2.2	SEQUENCE_RUN_MULTIPLE	Extension	787
10.23.2.3	BEFORE_AND_AFTER	Extension	787
10.23.2.4	SEQUENTIAL_ALTERNATES	Extension	788
10.23.2.5	SEQUENTIAL_ALTERNATES_KEEP_BEST	Extension	789
10.23.2.6	PROPORTIONAL_RESOLVES	Extension	790
10.23.2.7	ORDERED_RESOLVES	Extension	790
10.23.3	problem_selection extensions of model Active_Strategy	790	
10.23.3.1	PROPORTIONAL_INTERACTION	Extension	790
10.23.3.2	EARLIEST_PROBLEM_START	Extension	791
10.23.3.3	SORT_BY_EXPRESSION	Extension	791
10.24	Active_Goal Extensions	792	
10.24.1	goal extensions of model Active_Goal	792	
10.24.1.1	FEASIBILITY	Extension	792
10.24.1.2	MINIMIZE_PROBLEM_COUNT	Extension	792
10.24.1.3	MINIMIZE_PROBLEMS	Extension	792
10.24.1.4	MINIMIZE_LATENCY	Extension	793
10.24.1.5	WEIGHTED_LATENCY	Extension	793
10.24.1.6	WEIGHTED_SHORTNESS	Extension	793
10.24.1.7	MINIMIZE_SHORTNESS	Extension	793
10.24.1.8	MINIMIZE_COST	Extension	794
10.24.1.9	MAXIMIZE_PROFIT	Extension	794
10.24.1.10	MAXIMIZE_REVENUE	Extension	794
10.25	Item Extensions	796	
10.25.1	spec extensions of model Item	796	
10.25.1.1	STANDARD	Extension	796
10.25.1.2	CUSTOM	Extension	796
10.26	Skill Extensions	797	
10.26.1	selection extensions of model Skill	797	
10.26.1.1	PRIMARY	Extension	797
10.26.1.2	PREFER_PRIMARY	Extension	797
10.26.1.3	EVEN	Extension	797
10.26.1.4	MAX_EFFICIENCY	Extension	797
10.27	Skill_Resource Extensions	798	
10.27.1	efficiency extensions of model Skill_Resource	798	
10.27.1.1	FIXED	Extension	798
10.27.1.2	CALENDAR	Extension	798
10.28	Configuration Extensions	799	
10.28.1	spec extensions of model Configuration	799	
10.28.1.1	STANDARD	Extension	799
10.28.1.2	CUSTOM	Extension	799
10.29	Operation_State Extensions	800	
10.29.1	identifier extensions of model Operation_State	800	
10.29.1.1	EARLIEST	Extension	800
10.29.2	static spec extensions of model Operation_State	800	
10.29.2.1	STARTED	Extension	800
10.29.2.2	COMPLETED	Extension	801
10.29.2.3	IN_FRONT	Extension	801
10.30	Request Extensions	803	

Contents

10.30.1	delivery_policy extensions of model Request	803
10.30.1.1	FIXED Extension	803
10.30.2	delivery_naming extensions of model Request	803
10.30.2.1	NUMBERED Extension	803
10.31	Delivery_Request Extensions	803
10.31.1	promising_policy extensions of model Delivery_Request	804
10.31.1.1	BUCKETED_ALL Extension	804
10.31.1.2	BUCKETED_ASAP Extension	804
10.31.1.3	ON_TIME Extension	804
10.31.1.4	ALL Extension	804
10.31.1.5	ALL_ON_TIME Extension	804
10.31.1.6	ASAP Extension	804
10.31.1.7	ASAP_MONTHLY Extension	805
10.31.1.8	BUCKETED_ALLOCATION Extension	805
10.31.1.9	BUCKETED_ALL_MIN_PRICE Extension	805
10.31.1.10	BUCKETED_MIN_PRICE_ASAP Extension	806
10.31.1.11	SHIP_IN_RATIO Extension	807
10.31.2	fulfillment_policy extensions of model Delivery_Request	807
10.31.2.1	ON_TIME Extension	807
10.31.2.2	FULL_QUANTITIES_OF_ALL_ITEMS Extension	807
10.31.2.3	UNRESTRICTED Extension	807
10.32	Promise Extensions	808
10.32.1	delivery_policy extensions of model Promise	808
10.32.1.1	FIXED Extension	808
10.33	Delivery_Promise Extensions	809
10.33.1	fulfillment_policy extensions of model Delivery_Promise	809
10.33.1.1	ON_TIME Extension	809
10.33.1.2	FULL_QUANTITIES_OF_ALL_ITEMS Extension	809
10.33.1.3	UNRESTRICTED Extension	809
10.34	Horizon Extensions	810
10.34.1	bucket_size extensions of model Horizon	810
10.34.1.1	ONE Extension	810
10.34.1.2	MONTHS Extension	810
10.34.1.3	WEEKS Extension	810
10.34.1.4	DAYS Extension	810
10.34.1.5	DATES Extension	810
10.35	Delivery_Acceptance Extensions	812
10.35.1	fulfillment_policy extensions of model Delivery_Acceptance	812
10.35.1.1	ON_TIME Extension	812
10.35.1.2	FULL_QUANTITIES_OF_ALL_ITEMS Extension	812
10.35.1.3	UNRESTRICTED Extension	812
10.36	Strategy Extensions	813
10.36.1	termination extensions of model Strategy	813
10.36.1.1	NO_PROBLEMS Extension	813
10.36.1.2	TARGET_ACHIEVED Extension	813
10.36.1.3	RESOLVE_COUNT_EXCEEDED Extension	813
10.36.1.4	MANUAL Extension	813
10.36.2	execution extensions of model Strategy	813
10.36.2.1	SEQUENCE_RUN_ONCE Extension	813
10.36.2.2	SEQUENCE_RUN_MULTIPLE Extension	814

Contents

10.36.2.3	BEFORE_AND_AFTER Extension	814
10.36.2.4	SEQUENTIAL_ALTERNATES Extension	815
10.36.2.5	SEQUENTIAL_ALTERNATES_KEEP_BEST Extension	816
10.36.2.6	PROPORTIONAL_RESOLVES Extension	817
10.36.2.7	ORDERED_RESOLVES Extension	817
10.36.3	problem selection extensions of model Strategy	818
10.36.3.1	PROPORTIONAL_INTERACTION Extension	818
10.36.3.2	EARLIEST_PROBLEM_START Extension	818
10.36.3.3	SORT_BY_EXPRESSION Extension	818
10.37	Problem_Set Extensions	820
10.37.1	category extensions of model Problem_Set	820
10.37.1.1	REQUEST_NOT_PLANNED Extension	820
10.37.1.2	REQUEST_PLANNED_LATE Extension	820
10.37.1.3	REQUEST_PLANNED_EARLY Extension	821
10.37.1.4	REQUEST_PLANNED_SHORT Extension	821
10.37.1.5	REQUEST_PLANNED_EXCESS Extension	822
10.37.1.6	PROMISE_NOT_PLANNED Extension	822
10.37.1.7	PROMISE_PLANNED_LATE Extension	823
10.37.1.8	PROMISE_PLANNED_EARLY Extension	824
10.37.1.9	PROMISE_PLANNED_SHORT Extension	824
10.37.1.10	PROMISE_PLANNED_EXCESS Extension	825
10.37.1.11	ACCEPTANCE_NOT_PLANNED Extension	825
10.37.1.12	ACCEPTANCE_PLANNED_LATE Extension	826
10.37.1.13	ACCEPTANCE_PLANNED_EARLY Extension	827
10.37.1.14	ACCEPTANCE_PLANNED_SHORT Extension	827
10.37.1.15	ACCEPTANCE_PLANNED_EXCESS Extension	828
10.37.1.16	PLANNED_BEFORE_CURRENT Extension	829
10.37.1.17	UNRELEASED Extension	829
10.37.1.18	NEEDS_RELEASE Extension	830
10.37.1.19	INCONSISTENT_OPLAN Extension	830
10.37.1.20	OPERATION Extension	830
10.37.1.21	REQUEST_PROMISED_LATE Extension	831
10.37.1.22	REQUEST_PROMISED_EARLY Extension	831
10.37.1.23	REQUEST_PROMISED_SHORT Extension	832
10.37.1.24	REQUEST_PROMISED_EXCESS Extension	833
10.37.1.25	PROMISE_NOT_OFFERED Extension	834
10.37.1.26	PROMISE_NOT_ACCEPTED Extension	834
10.37.1.27	ACCEPTANCE_INCONSISTENT Extension	834
10.37.1.28	REQUEST_QUEUED Extension	835
10.37.1.29	REQUEST Extension	835
10.37.1.30	REQUEST_PLAN Extension	836
10.37.1.31	PROMISE Extension	836
10.37.1.32	PROMISE_PLAN Extension	837
10.37.1.33	DELIVERY_REQUEST_EXTENSION Extension	837
10.37.1.34	DELIVERY_REQUEST_NOT_COORDINATED Extension	838
10.37.1.35	DELIVERY_PROMISE_NOT_COORDINATED Extension	838
10.37.1.36	DELIVERY_ACCEPTANCE_NOT_COORDINATED Extension	838
10.37.1.37	SUPPLY_PLANNED_LATE Extension	839
10.37.1.38	SUPPLY_PLANNED_EARLY Extension	839
10.37.1.39	SUPPLY_PLANNED_SHORT Extension	840

Contents

10371.40	SUPPLY_PLANNED_EXCESS Extension	840
10371.41	SUPPLY_PROMISED_LATE Extension	841
10371.42	SUPPLY_PROMISED_EARLY Extension	841
10371.43	SUPPLY_PROMISED_SHORT Extension	842
10371.44	SUPPLY_PROMISED_EXCESS Extension	843
10371.45	SUPPLY_PLAN Extension	843
10371.46	SUPPLY_PLAN Extension	844
10371.47	SUPPLY_PROMISE Extension	844
10371.48	UNIDENTIFIED_OF_STATE Extension	845
10371.49	UNCONSOLIDATED Extension	845
10371.50	UNCOORDINATED Extension	846
10371.51	CONSOLIDATION_OVERSIZE Extension	846
10371.52	CONSOLIDATION_UNDERSIZE Extension	846
10371.53	OVERLOAD Extension	847
10371.54	OVERSIZE Extension	847
10371.55	BUCKET_OVERSIZE Extension	848
10371.56	UNDERLOAD Extension	848
10371.57	RESOURCE Extension	848
10371.58	NEGATIVE_ON_HAND Extension	849
10371.59	OVER_FLOW_LIMIT Extension	849
10371.60	NEGATIVE_ON_HAND_AT_END Extension	850
10371.61	LOT_OVER_CONSUMED Extension	851
10371.62	LOT_NOT_CONSUMED Extension	851
10371.63	LOT_NOT_PRODUCED Extension	851
10371.64	LOT_OVER_PRODUCED Extension	852
10371.65	LOW_ON_HAND Extension	852
10371.66	EXCESS_ON_HAND Extension	853
10371.67	EXCESS_ON_HAND_AT_END Extension	854
10371.68	BUFFER Extension	854
1038	Strategy_Change Extensions	855
1038.1	change category extensions of model Strategy_Change	855
1038.1.1	MOVE_IN Extension	855
1038.1.2	MOVE_OUT Extension	855
1038.1.3	SPLIT Extension	855
1038.1.4	USE_MORE Extension	855
1038.1.5	USE_LESS Extension	855
1038.1.6	USE_ALTERNATE_OPERATION Extension	855
1038.1.7	USE_ALTERNATE_RESOURCE Extension	855
1038.1.8	USE_EFFECTIVE_ALTERNATE Extension	856
1038.1.9	INCREASE_CAPACITY Extension	856
1038.1.10	DECREASE_CAPACITY Extension	856
1038.1.11	RESIZE_MORE Extension	856
1038.1.12	RESIZE_LESS Extension	856
1038.1.13	UPSTREAM Extension	856
1038.1.14	DOWSTREAM Extension	856
1039	Strategy_Lock Extensions	857
1039.1	spec extensions of model Strategy_Lock	857
1039.1.1	OPERATION_PLAN_RANK_RANGE Extension	857
1039.1.2	OPERATION_PLAN_RANK_EXPRESSION Extension	857
1040	Strategy_Goal Extensions	858

Contents

1040.1	goal extensions of model Strategy_Goal	858
1040.1.1	FEASIBILITY Extension	858
1040.1.2	MINIMIZE_PROBLEM_COUNT Extension	858
1040.1.3	MINIMIZE_PROBLEMS Extension	858
1040.1.4	MINIMIZE_LATENESS Extension	859
1040.1.5	WEIGHTED_LATENESS Extension	859
1040.1.6	MINIMIZE_SHORTNESS Extension	860
1040.1.7	WEIGHTED_SHORTNESS Extension	860
1040.1.8	MINIMIZE_COST Extension	860
1040.1.9	MAXIMIZE_PROFIT Extension	860
1040.1.10	MAXIMIZE_REVENUE Extension	861
1041	Calendar_Empty Extensions	862
1041.1	value extensions of model Calendar_Empty	862
1041.1.1	NUMBER Extension	862
1041.1.2	QUANTITY Extension	862
1041.1.3	NUMBER_QUANTITY Extension	862
1041.1.4	SYMBOL Extension	862
1041.1.5	TIME Extension	862
1041.2	day pattern extensions of model Calendar_Empty	863
1041.2.1	EVERYDAY Extension	863
1041.2.2	EVERY_N-DAYS Extension	863
1041.2.3	WEEKDAYS Extension	863
1041.2.4	WEEKENDS Extension	863
1041.2.5	DAYS_OF_WEEK Extension	863
1041.2.6	DAYS_OF_MONTH Extension	864
1041.2.7	DAY_OF_WEEK_OF_MONTH Extension	865
1041.2.8	DAY_OF_LAST_WEEK_OF_MONTH Extension	866
1041.2.9	DAY_OF_YEAR Extension	866
1041.2.10	YEARLY Extension	867
1042	Calendar Extensions	868
1042.1	empty value extensions of model Calendar	868
1042.1.1	UNSPECIFIED Extension	868
1042.1.2	NUMBER Extension	868
1042.1.3	QUANTITY Extension	868
1042.1.4	NUMBER_QUANTITY Extension	868
1042.1.5	SYMBOL Extension	869
1042.1.6	TIME Extension	869
1043	Flow_Criterion Extensions	870
1043.1	criterion extensions of model Flow_Criterion	870
1043.1.1	CUSTOMER_RANK Extension	870
1043.1.2	SELLER_RANK Extension	870
1043.1.3	REQUEST_RANK Extension	870
1043.1.4	ACTUAL_OR_FORECAST Extension	871
1043.1.5	REQUEST_ISSUED Extension	871
1043.1.6	PROMISE_DUE Extension	872
1043.1.7	REQUEST_DUE Extension	872
1043.1.8	DUE_DATE Extension	873
1043.1.9	PROMISED Extension	873
1043.1.10	ENTRY_DATE Extension	874
1044	Calendar_Plan Extensions	875

Contents

10.44.1	envy_value extensions of model Calendar_Plan	875
10.44.1.1	NUMBER Extension	875
10.44.1.2	QUANTITY Extension	875
10.44.1.3	NUMBER_QUANTITY Extension	876
10.44.1.4	SYMBOL Extension	877
10.44.1.5	TIME Extension	878
10.45	Formal Extensions	879
10.45.1	spec extensions of model Formal	879
10.45.1.1	Void Extension	879
10.45.1.2	Logical Extension	879
10.45.1.3	String Extension	880
10.45.1.4	Symbol Extension	880
10.45.1.5	Date Extension	881
10.45.1.6	Date_Range Extension	883
10.45.1.7	Number Extension	883
10.45.1.8	Percentage Extension	885
10.45.1.9	Integer Extension	886
10.45.1.10	Quantity Extension	889
10.45.1.11	Quantity_Range Extension	890
10.45.1.12	Time Extension	891
10.45.1.13	Restriction Extension	892
10.45.1.14	Horizon Date Extension	893
10.45.1.15	List Extension	895
10.46	Field Extensions	897
10.46.1	field_type extensions of model Field	897
10.46.1.1	SIMPLE Extension	897
10.46.1.2	SELECTOR Extension	897
10.46.1.3	EXTENDED Extension	897
10.46.1.4	USER Extension	897

1 Introduction

1.1 Preface

You may wish to skip Chapter 1, Chapter 2 (which introduces the basic types), and Chapter 3 (which discusses basic worksheet functions), and jump immediately into the Supply Chain model in Chapter 4. You can return to the basic types and functions as you encounter them in the models. You may need to return to this Introduction for a quick overview of OIL. Expressions if you find examples to be unclear, or for a quick overview of the other manuals available to you.

However, most will benefit from at least reading this Introduction, even if you choose to skip the basic types and functions in chapters 2 and 3.

1.2 Overview

RHYTHM Supply Chain Planner (SCP) is an innovative decision support system designed to give supply chain decision makers unprecedented visibility of their supply chain and intelligent suggestions on how to better plan it. It is designed to work cooperatively with all the decision makers in the supply chain (the planners, managers, sales personnel, etc.) to both be more effective in their domain and to be more effective with the other decision makers involved. As such it must be customizable to the needs of each of the varied decision makers in the supply chain. SCP is designed to be easy to customize by its users (planners, managers, salespeople -- not programmers).

SCP is a Truly Integrated Planning (TIP) system. There are many books, papers, and advertising on the importance of "integrated planning". However, the "integrated" solutions that are presented are consistently dis-integrated. The solutions either involve getting dis-integrated planning packages to communicate, or bringing dis-integrated planning tools into one package. Putting one user interface over a set of tools and one database under a set of tools does not integrate the planning performed by the independent tools. And it does not provide the decision makers with the visibility and capabilities needed to perform Truly Integrated Planning.

Truly Integrated Planning involves integration of the material and capacity planning, integration of factory and distribution planning, integration of master and operational (execution) planning, integration of manual and automated planning, integration of make-to-stock and make-to-order planning, integration of discrete and repetitive planning, and more.

Effective supply chain management depends upon the ability to have integrated plans throughout the supply chain. Supply chains are typically diverse in their needs. Repetitive suppliers

will feed discrete manufacturers. An essentially make-to-stock facility will be fed by a make-to-order facility and will feed standard components to an assemble-to-order packaging plant. The key to success is at the customer interface, so no matter how well the manufacturing facility performs, if the distribution plan is faulty the manufacturer will suffer. If the key to success is in satisfying their customers, the decision makers must have visibility of the complete supply chain responsible for that satisfaction in order to make the decisions that will optimize that satisfaction.

For a more thorough discussion of Truly Integrated Planning and the other concepts on which SCP is based, see the *RHYTHM Supply Chain Planner Concept Manual*.

1.3 RHYTHM Manuals

The *RHYTHM SCP Concept Manual* describes the basic concepts on which SCP is based. It discusses Truly Integrated Planning (TIP), advanced supply chain management with TIP, customization to support TIP, and many of the other innovative features that TIP demands and RHYTHM SCP provides.

This manual, the *RHYTHM SCP Model Reference Manual*, describes the models that the user can set up and access via the Interactive Reports. Each model consists of fields, and the fields are functions in the Report's spreadsheet-like expression language. This information is available through online help.

The models and fields define all the functionality that can be accessed by the Interactive Reports, and thus almost all the functionality available from the RHYTHM product suite.

The *RHYTHM SCP Customization Manual* provides the tools and information necessary to perform the tasks involved in customizing RHYTHM SCP. The manual consists of reference materials designed to supplement the RHYTHM SCP Model Reference Manual. This manual is also available as online help.

The *RHYTHM SCP User Manual* provides users with basic conceptual information, how to information, and detailed information for each of the workbenches, menus, and windows within the interface. This manual is only available as online help and is accessed from the Help Topics option of the Help menu.

The *RHYTHM SCP Q&A Manual* is designed around "How do I use SCP to do X?" questions. It is designed to cover commonly asked questions. It is designed to map common scenarios with other paradigms to SCP. It is designed to present challenging modeling or usage problems and illustrate one or more solutions. It is designed to help show the flexibility and power of the mechanisms by presenting a variety of problems and showing a variety of solutions.

1.4 Modeling

1.4.1 Expressions

Expressions, the Object Interaction Language, and all of their applications in the Interactive Reports are discussed in detail in the *RHYTHM SCP Customization Manual*. However, to keep this manual stand-alone, a brief introduction to Expressions is appropriate.

OIL Expressions are very much like traditional spreadsheet expressions. An expression evaluates to a data value. There are various functions which can take one or more data values as parameters and compute a new data value.

1.4.2 Types

Each data value in a model has a data "type". For instance, two common types are 'Number' and 'String'. The functions take parameters with specific types and evaluate to a specific type of data value. Formatting of data for display is type-specific. Controls for interacting with and editing data values are type-specific. Help is type-specific. And so on.

There are a number of basic types, such as 'Number', 'Quantity', 'Date', 'Time', and 'String'. All the basic types are described in chapter 2.

There is also a List type. The List type holds any number of elements, where each of the elements is a data value of the same type. The type of the data elements in the List is specified within brackets [], such as List[Number] or List[Date].

Beyond the typical data types, each model described in this manual is a "model type" in the Object Interaction Language. For example, 'Operation', 'Resource', 'Buffer', 'Product', 'Site', 'Supply Chain', and 'Plant' are all types; specifically, model types. The term "model" is used to mean an instance/value of a "model type" (for object-oriented programmers, this equates nicely to "objects" which are instances of a "class"). For example, the DRILL32 may be a model of the Resource model type. So, an Expression can compute a Resource, DRILL32, which may be passed as a parameter to another function in the Expression to compute Calendar for that Resource.

Finally, Expressions are themselves a data type in the Object Interaction Language. Thus, Expressions can evaluate to other Expressions, and fields of models can hold Expressions which are evaluated by the planning engine to compute values dynamically. That provides programmability in a form that is identical to writing Interactive Reports, and thus familiar to most TRIPS users. (In fact it is familiar to most spreadsheet users.)

1.4.3 Fields

In addition to the functions typically found in traditional spreadsheets, each of the fields in this manual are functions in the Object Interaction Language. In fact, that is the purpose of the

Object Interaction Language: to give flexible access to the data and functionality in the models that are being created, edited, and planned.

The typical field is a function that takes a model (a data value of a model type) as an argument and returns a particular data value out of that model. For example, the 'name' field function will take a Resource as an argument and will return the Symbol that is the name or id of that Resource; the 'location' field function will take a Resource and return a Location which models the location of that Resource.

Fields can take additional arguments. For example, the 'load_time' field takes a Resource, Plan and a Date_Range. It returns the hours (Time) of load that is planned on that Resource during that Date_Range.

Some fields are view-only. That is, they are not scutable by the user. They are analysis outputs, not user inputs. View-Only is just one of the possible properties of a field which are documented after the field description.

All the modeling and planning functionality provided by SCP is provided through field functions described in this manual. Almost all windows and reports available in SCP were created with Expressions and constructs of the Object Interaction Language, using field functions that can be found in the *RHYTHM SCP Model Reference Manual*. Thus, most anything you see in SCP, you can customize or wholly rewrite.

1.4.4 Models

This manual describes the models that can be used to model your supply chain, its current state, and plans for it. For example, there are models for Operations, Resources, Buffers, Products, Sites, Supply_Chains, Plans, and Problems.

A model is essentially defined by the fields that make it up. A field defines a data value that describes a characteristic of the real-world thing being modeled. For example, the location of a machine; the machine is modeled as a Resource; the location is modeled as a Location; and the Resource has a field 'location' that specifies the Location of that Resource.

1.4.5 Submodels

The SCP models form a hierarchy. For example, a Supply_Chain model contains any number of Site models, Seller models, and Plan models. Site models contain Operation, Resource, and Buffer models. Operation is a submodel of Site, which is a submodel of Supply_Chain.

Each submodel has an 'owner' field which returns the model that it is a submodel of. And that owning model has a field 'resources' which is a list of that submodel type containing each submodel owned by it.

For example, Resource's 'owner' field returns the Site that owns that Resource. And Site has a field 'resources' which contains all the Resource models that it owns.

Many models have a key field, a unique identifier for that model within its owner. For instance, Resource's key field is 'name'. Each Resource in a Site must have a unique name. And given a Site and a name, it is possible to find the Resource model with that name.

Except for the key field, all fields of a model will default to some meaningful value if a value is not supplied. The default values for each field are documented after the field description. This defaulting eases creation of new models.

1.4.6 Extensions

The SCP models are based upon our Extensible Model Architecture™, which is a unique approach to modeling that allows a model to be extended with additional fields depending upon the value of a particular field.

In this way, a simple model can be provided to support the majority of scenarios. But for the few instances that require much more modeling sophistication, the model can be extended with additional information and additional semantics, without imposing cost on the rest of the models.

Similarly, a single product (RHYTHM SCP) can provide all of the functionality needed to model the various elements that make up a supply chain without burdening the models of users that do not need all that different functionality. In that way, different users only have to deal with what they want to deal with.

A field identified as an extension selector specifies which extension to use for a particular semantic. Extensions define the semantic behavior of the model when that extension is added, and the additional fields that are made available on that model.

1.4.7 User-Defined Fields

Some model types can be extended by the addition of user-defined fields, which can be of any data type specified by the user. Such user-defined fields are available for use as functions in expressions as if they were built-in fields. The planning algorithms can even be parameterized such that they read the user-defined fields in performing planning actions.

Not all model types can be extended in this way. Some model types are very light-weight components of other models. As such, allowing user-defined fields on each of the components would be quite expensive (computationally). Some models are transient in nature. Allowing user-defined fields would be somewhat deceiving since they would disappear at the whim of the algorithms.

1.5 Manual Organization

1.5.1 HyperText

This is a hypertext manual. Hyperlinks are printed in bold. Most type names are links, most model names are links, most field functions are links, and so on. You can utilize these links by viewing with FrameMaker, FrameViewer, or an HTML (WWW) Browser (such as Mosaic).

1.5.2 Notation

Type names (and thus model type names) are written capitalized (e.g., Supply, Chan). Field function names are written in single quotes, as are Expressions (e.g., 'sites' or 'A1 + B2'). Data values are written in double quotes (e.g., "36 gal"). Extension names are typically in all capital letters (e.g., PRODUCE_YIELD_CALENDAR).

1.5.3 Chapters

The next chapter discusses all the Basic Types. The third chapter defines the Basic Functions, including many of those that a spreadsheet user would find familiar. The fourth chapter defines all the models. The fifth chapter provides the catalog of extensions that can be chosen. An index follows.

1.5.4 Fields

Each field is documented with a consistent, stand-alone block, which is suitable as well for on-line Help display. The redundant information from a manual point-of-view is in a standard header line that states the type of the field, whether it is a field, submodel, or extension, and the model that contains it.

The description of that field follows. After the description, the default value is named and then if there are any properties (such as View-Only), they are documented.

Extension selector fields will each be followed by a list of all the extensions, providing a mini-index to those extension descriptions.

1.5.5 Models

The models are documented in chapters, hierarchically. Submodels are documented in subchapters of their 'owner' model.

The model header gives the name and the name of the 'owner' model. That is followed by the description of that model and how it is used to model real world things.

Various summaries are then provided, such as what submodels it contains, and what extensions. Since these are links, it provides mini-indexes per model chapter.

That is followed by each of the field descriptions as described previously.

Then in subchapters come each of the submodels of that model. Also in subchapters is the submodels of extensions.

1.5.6 Extensions

The extensions are documented in the Extensions chapter, which is organized with a subchapter per model, and a sub-subchapter per extension selector of that model.

Each extension header gives its name, the model that it is an extension of, and the extension selector used to select it. That is followed by a description of the semantics that the extension adds to its model. Finally, descriptions of the fields that are added to the model by that extension are described, as discussed previously.

Introduction

2 Basic Types

2.1 Cell_State

Cell State

2.2 Computed_String

This is an obsolete type that will soon disappear. All uses of Computed_String will become simply String.

For now, it distinguishes a "computed" String, one that is not just a view of a String that lives elsewhere, but rather actually holds the characters.

2.3 Date

A particular point in history (past or future), with precision of one second. For example, the Date "1989 Dec 19 12:50:32". A Date must be in the range 1970 to 2099. A Time can be added to a Date, producing a Date that much Time in the future. Two Dates can be subtracted resulting in a Time that is equal to the duration between those two Dates.

The Dates "infinite_past" ("----") and "infinite_future" ("++++") are also supported, with normal infinite semantics (adding to or subtracting from an infinite Date, results in the same infinite Date; subtracting two infinite Dates is an error).

Note that in some systems the "date" fields just model the day information, and separate "time-of-day" fields give the time within that day. This is simple for formatting and purely database-ish applications. However, such separation is illogical and cumbersome for systems that allow sophisticated computations on the Dates. Thus, a Date models all that is needed to "make a date with someone", both the day and the time-of-day. And Time models duration (e.g., number of hours).

All times are stored internally as seconds from some particular time in the past. So, there is no hard coding of centuries, etc. Sorting is done before converting the date to text, so the sequence will always be correct, even if the date format does not include the year at all. The sort before converting to text works for quantities, too. For example, "1 yr" is bigger than "20 day". Specific conditions that are satisfied include:

- * 20th century dates in DD/MM/YY format entered retrospectively once the year 2000 is reached are interpreted correctly as 20th century dates, e.g. 12/05/98.
- * Dates in DD/MM/YY format entered in the 20th century for future dates in the 21st century are interpreted correctly as being the 21st century dates, e.g. 01/03/01.

* Data queries that cross the millennium, e.g. between 1998 and 2002 return the correct data, instead of an error or no data.

When parsing any data format which has a 2 digit year, years less than 70 are assumed to be in the 21st century. For example, 01-01-01 with format YY-MM-DD is January 1, 2001.

2.4 Date_Range

A range of Dates from a 'start' Date up to (but not including) an 'end' Date. Functions 'start' and 'end' can get and set the bounds of a Date_Range. Beyond being just a pair of Dates, a Date_Range supports special formatting. For example, a Date_Range can be displayed or input as "July 1991", "Week 12 1993", "3Q 1995", or "95-02-12/95-04-15".

A Date_Range can be created from two Dates 'date_range(date1, date2)' or from a start Date and a Time 'date_range(start, time)'.

There are also special functions to generate lists of consecutive Date_Ranges, which is perhaps the primary purpose for the Date_Range type. For example, 'months(start, end)' returns a List of Date_Ranges (List(Date_Range)) that start and end on month boundaries. Similarly, 'days', 'weeks', 'quarters', and 'years' are available. Combinations of those can be formed by using the 'list' function to combine lists from different of those functions. For example, 'list(weeks(start, start+4 weeks), months(start+4 weeks, end))' provides 4 weeks followed by several months.

2.5 Event

A User interaction or other occurrence that should be "handled" by a GUI client. For example, a key press or release, or a mouse-button press, drag, or release.

2.6 Expression

The value is a expression that can be evaluated to compute something. The expression syntax is the same as is available in the Report Worksheet.

2.7 Func

This type is obsolete.

The value is a function which can be passed into other functions, called and executed. It is used in particular by 'List.for_each(function)', which executes the function for each element of the List.

2.8 Horizon_Date

Horizon_Date is a Date specified relative to the start of the planning horizon. Thus, as time passes, and the planning horizon moves forward, so do these Dates.

Horizon_Date is used primarily to specify time fences, which are necessarily horizon-relative.

It allows fairly sophisticated specification of relative dates, such as the second Monday, the first day of the third month, the fourth Thursday of the second month, and so on.

It has minute granularity: so you can specify a Date at 8:30am, but you cannot specify a Date at 8:30:22am.

Currently, **OLL** treats the time part of the horizon date not as an offset, but as a specific time of day. So, if the horizon date is listed as "1 day at 11:00", the resulting horizon will be the next day at 11:00. Because time is absolute and not relative, you can not have a horizon date that is only a time. The time component of a horizon date is optional. If not specified, a time of 00:00 is assumed.

2.9 ID

Used by charts to send unique identifiers to the GUI. There's an automatic conversion from any model-type to ID.

2.10 Integer

An integer (number with no fractional part) in the range from 2 billion down to -8 million (approximately). This is typically used for small whole numbers that specify the number of things, or which one of that number of things (e.g., the 15th of 240). Using Numbers for that allows absurdity (e.g., 15.74th of 240.381).

2.11 Logical

A truth value: either true or false.

2.12 Measure

A physical measure or dimension. For example, mass, length, time, money, temperature, and piece-count are all base Measures. Measures can be composed from the base measures. For example, volume is length[3], area is length[2], velocity is length*time[-1], force is mass*length*time[-2], and money*time[-1], mass*time[-1], and so on are all possible.

There are 8 base Measures. Time and Money are predefined an immutable. The other 6 can be largely redefined as most useful for the manufacturer. For instance, temperature may have no meaningful usage for a manufacturer, so he can effectively use it to measure in "number of cans".

2.13 Measure_Unit

A **Measure_Unit** is a Measure plus preferred units in which to display a Quantity with that Measure. It is used primarily to specify how to convert Quantities. A Quantity may be converted to use different units of the same Measure; or may be converted to a different unit and Measure of a particular Item, Operation, etc.

For example, mass, length, volume, time, velocity, piece-count, and rate are all Measures. A Quantity with Measure mass can be displayed or input in various units; for example, "kg", "g", "pounds", or "tons". Similarly, length could be in "km", "cm", "miles", or "ft"; time could be in "s", "min", "hr", "days", or "weeks"; velocity could be in "km/hr" or "ft/s". And so on.

One unit of a particular Item may be defined to be "100 kg", "50 liters", and "\$750". Thus, a unit-of-an-Item is defined in terms of one or more unit-of-Measure. A Quantity of "400 kg" of that Item may then be converted to Measure_Unit "\$" (which would be "\$3000") or to Measure_Unit "ml" (which would be "200000 ml").

The base units which can be used by **Measure_Unit** for a particular Measure are defined by the **Measure_Base** models. **Measure_Base** defines that "hr" is "60 min", and that "min" is "60 s". **Measure_Base** models can be defined by the user.

Note that **Measure_Base** defines a conversion from one unit of a Measure to another unit of the same Measure. Such Measures, units, and conversions apply to anything. The Unit model defines a conversions between units of different Measures that are specific to an Item, Buffer, Operation, or Product. Such Units and conversions are not generic like units of Measures.

2.14 Money

Money is effectively a special case of Quantity. It is a Quantity with Measure "money", a currency value such as "\$13" or "4800 yen". It is separated out because of its importance in representing costs, and its special needs for "native" currency at different Sites. It can be easily intermixed in expressions with Quantity's, but where Money is called for, the Measure of the Quantity must be simply "money".

Note that Money formatting can be different than normal Quantities.

2.15 Number

A real number (can have fractional part) in the range from 1e37 (1 with 37 0's) down to -1e37 (approximately). It can represent numbers as small in magnitude as 1e-45 (decimal point, 45 0's, 1).

However, it has only limited precision: approximately 6 decimal digits.

So, $100,000,000 + 100 = 100,000,096$

$100,000,000 + 1 = 100,000,000$ (no change) Another example: typing 123456789 will result in 123456792 (only the first 7 digits are accurate). Luckily, there are very few applications that require more than 6 digits of precision.

2.16 Pathname

Specifies the name of a file or directory in the host's file system.

2.17 Percentage

Percentage is simply a Number. Mathematically it is used just like any Number. It is formatted the same as any Number. It freely converts to a Number, and Numbers can be freely converted to Percentages.

The only difference is that the value is typically displayed in percentage form ("50%") rather than normal form ("0.5"). By defining such fields to have a Percentage type rather than Number type, the user can set up different default Formats for all the Percentage fields.

Note that inputting "1" is equivalent to "100%", and "100" is equivalent to "10000%".

2.18 Predicate

A variable with type Predicate specifies a predicate descriptor.

2.19 Priority

A function execution priority.

2.20 Property_List

This is an internal type and should not be visible to users.

A list of properties is used to allow users to extend models by adding their own fields.

2.21 Quantity

An amount of something, specified by a Number and a Measure_Unit. For instance, "32 hr", "15 ft", "22 kg", "\$3620", and "16" are all Quantities. The last is a unitless Quantity, which is freely convertible to a simple Number.

The convert functions allow a Quantity to be converted from one Measure_Unit to another (the Measures must be the same). For instance, convert("kg", "4.4 lbs") will result in "2 kg". Quantities that are combined together will tend to combine or merge the Measure_Units into the result. For instance, "33 kg" / "3 hr" will result in "11 kg/hr".

Note that "0" of any Measure can be converted to "0" of any other. Also, convert will invert a number to get a match. For instance, "10 unit/hr" can be converted to "hr/unit". It will be "0.1 hr/unit". Or it can be converted to "min/unit", which will give "6 min/unit". This ability to invert on conversions can greatly ease use of rates.

2.22 Quantity_Range

A range of Quantity from a 'min' Quantity up to a 'max' Quantity, inclusive. Functions 'min' and 'max' can get and set the bounds of a Quantity_Range. Beyond being just a pair of Quantity's, a Quantity_Range supports special formatting. For example, a Quantity_Range can be displayed or input as "[2.5; 7.5]", "2.5 + 5", "5 + 2.5", or "2.5 to 7.5".

A Quantity_Range can be created from two Quantity's quantity_range(q1, q2).

2.23 Record

A variable with type Record specifies a RhythmLink data record.

2.24 Reference

An indirect pointer to a value which may be in a worksheet cell or a variable. There is an automatic conversion from Cell_State to Reference.

2.25 Restriction

A Restriction restricts a Date or Date_Range to a certain set of values. It can restrict either the "start" or the "end" of a Date_Range, or the whole Date_Range. It can specify a Date_Range that the Date(s) must be "within" or "not within". Note that "start after <some Date>" actually functions as though specifying a start within date and a date in the infinite future. This means that the restriction restricts the date to a start time within the range between some date following the date specified and a date in the future. Also, "end before <some Date>" functions as though specifying an end time within the infinite past and the specified date. This means that the restriction restricts the end date to a range between some date in the future and the date specified. To specify infinite future, type "+", or "-". For example, setting the restriction to "start first in 97-10-15 00:00 / +" will immediately change this hint to "start after 97-10-15 00:00" in the display. An empty Restriction "" will cause an error. However, it is possible to specify [], which automatically changes to [unspecified] (the default value).

Restrictions are commonly used to temporarily guide the automated planning process (a hint). Such hints can be imposed by manual intervention. The engine then tries to follow it, and then the hint is done. The hint is no longer followed during any subsequent planning or strategy runs.

Note this is specifically a Restriction of a Date or Date_Range. There are many other hints and locks that "restrict" the plan in other ways, but those are not of type Restriction.

2.26 Safety_Stock_Units

An enumeration value used by the CALENDAR stocking_policy extension of the Buffer model, in the default_user_bias_type field. See that field for details.

2.27 String

A text string: simply a sequence of letters, numbers, punctuation, spaces, tabs, line-feeds, etc. It can be of any length.

Most types can be converted (or formatted) to and from a String.

2.28 Symbol

A String that is used as the name or identifier of a model or object. It can be freely converted to a String with no change, and used wherever a String can. A String can be freely converted to a Symbol, but leading and trailing whitespace (spaces, tabs, line breaks, etc.) will be trimmed off (leaving such characters on identifiers can lead to data errors that are hard to see and correct).

2.29 Time

Time is effectively a special case of Quantity. It is a Quantity with Measure Time, a duration such as "13 hr". It is separated out because of its importance, allowing it to be handled specially, and restricted appropriately (Times cannot be assigned Quantities such as "15ft"). It can be easily intermixed in expressions with Quantity's, but where a Time is called for, the Measure of the Quantity must be simply Time.

Note that Time formatting can be different than normal Quantities. In particular, the format "hh:mm:ss" is supported for Times. This is particularly useful when the Time field represents the time since the start of a day. Thus, adding the Date "91-07-23" to the Time "12:52:32" results in the Date "91-07-23 12:52:32", as would be expected.

Also note that in some systems the "time" models the time-of-day portion of a Date, and "date" models only the day information. This is simple for formatting and purely database-ish applications. However, such separation is illogical and cumbersome for systems that allow sophisticated computations on the Dates. Thus, a Date models all that is needed to "make a date with someone" both the day and the time-of-day. And Time models duration (e.g., number of hours). However, as noted above, where a Time field means "since the start of a day", a Time field can seem very much like other systems "time-of-day" fields: 12:52 means 12 hours and 52 minutes since the start of the day, but it also looks like a time-of-day in 24-hour format.

2.30 Type

A variable with type Type specifies some type, such as Quantity, Date, Symbol, or even Type itself.

2.31 Typed_Value

A pair, comprising a value and the type of the value. A Typed_Value may specify, for example, a model instance and the Model_Type* which describes it.

2.32 Void

No value is expected or returned.

3.1 Operators

3 Basic Functions

Table 1: Operators

Operator	Precedence	Parameters	Return Type	Description	Example
-	4	(Integer)	Integer	Integer negate	-9000
-	4	(Number)	Number	Floating negate	-10.2
-	4	(Quantity)	Quantity	Quantity negate	-52
-	4	(Time)	Time	Time negate	-021530
-	4	(Date, Date)	Time	Subtract two dates to get the time difference	11MAY1995000000 - 01MAY1995000000
-	4	(Date, Time)	Date	Subtract a time from a date	11MAY1995000000 - 021530 returns 10MAY1995214430
-	4	(Integer, Integer)	Integer	Subtract two integers	9000 - 1000 returns 8000
-	4	(Number, Number)	Number	Subtract two floating numbers	10.2 - 6.4 returns 3.8
-	4	(Quantity, Quantity)	Quantity	Subtract two quantities	52 - 36 returns 16
-	4	(Time, Time)	Time	Subtract two times to get the time difference	021530 - 010000 returns 011530
-	4	(Date, Time)	Date	Add a time to a date to get a new date	11MAY1995000000 + 021530 returns 11MAY1995021530
-	4	(Time, Date)	Date	Add a date to a time to get a new date	021530 + 11MAY1995000000 returns 11MAY1995021530
-	4	(Integer, Integer)	Integer	Add two integers	9000 + 1000 returns 10000
-	4	(Number, Number)	Number	Add two floating numbers	10.2 + 6.4 returns 16.6
-	4	(Quantity, Quantity)	Quantity	Add two quantities	52 + 36 returns 88
-	4	(Time, Time)	Time	Add two times	021530 - 010000 returns 031530
-	5	(Integer, Integer)	Integer	Multiply two integers	9000 * 1000 returns 9000000
-	5	(Number, Number)	Number	Multiply two floating numbers	10.2 * 6.4 returns 65.28

Table 1: Operators

Operator	Precedence	Parameters	Return Type	Description	Example
*	5	(Quantity, Quantity)	Quantity	Multiply two quantities	52 * 36 returns 1872
/	5	(Integer, Integer)	Integer	Divide two integers	9000 / 1000 returns 9
/	5	(Number, Number)	Number	Divide two floating numbers	10.2 / 6.4 returns 1.594
/	5	(Quantity, Quantity)	Quantity	Divide two quantities	52 / 36 returns 1.44
<	3	(Date, Date)	Logical	Less than date	11MAY1995000000 < 21MAY1995000000 returns TRUE
<	3	(Integer, Integer)	Logical	Less than integer	9000 < 1000 returns FALSE
<	3	(Number, Number)	Logical	Less than floating	10.2 < 6.4 returns FALSE
<	3	(Quantity, Quantity)	Logical	Less than quantity	52 < 36 returns FALSE
<	3	(String, String)	Logical	Less than string (case insensitive)	"ORDER_12" < "ORDER_22" returns TRUE
<	3	(Symbol, Symbol)	Logical	Less than symbol (case insensitive)	"ORDER_12" < "ORDER_22" returns TRUE
<	3	(Time, Time)	Logical	Less than time	021530 < 010000 returns FALSE
<=	3	(Date, Date)	Logical	Less than or equal date	11MAY1995000000 <= 21MAY1995000000 returns TRUE
<=	3	(Integer, Integer)	Logical	Less than or equal integer	9000 <= 1000 returns FALSE
<=	3	(Number, Number)	Logical	Less than or equal floating	10.2 <= 6.4 returns FALSE
<=	3	(Quantity, Quantity)	Logical	Less than or equal quantity	52 <= 36 returns FALSE
<=	3	(String, String)	Logical	Less than or equal string (case insensitive)	"ORDER_12" <= "ORDER_22" returns TRUE
<=	3	(Symbol, Symbol)	Logical	Less than or equal symbol (case insensitive)	"ORDER_12" <= "ORDER_22" returns TRUE
<=	3	(Time, Time)	Logical	Less than or equal time	021530 <= 010000 returns FALSE
>	3	(Date, Date)	Logical	Greater than date	11MAY1995000000 > 21MAY1995000000 returns FALSE

Table 1: Operators

Operator	Precedence	Parameters	Return Type	Description	Example
>	3	(Integer, Integer)	Logical	Greater than integer	9000 > 1000 returns TRUE
>	3	(Number, Number)	Logical	Greater than floating	10.2 > 6.4 returns TRUE
>	3	(Quantity, Quantity)	Logical	Greater than quantity	52 > 36 returns TRUE
>	3	(String, String)	Logical	Greater than string (case insensitive)	"ORDER_12" > "ORDER_22" returns FALSE
>	3	(Symbol, Symbol)	Logical	Greater than symbol (case insensitive)	"ORDER_12" > "ORDER_22" returns FALSE
>	3	(Time, Time)	Logical	Greater than time	021530 > 010000 returns TRUE
>=	3	(Date, Date)	Logical	Greater than or equal date	11MAY1995000000 >= 21MAY1995000000 returns FALSE
>=	3	(Integer, Integer)	Logical	Greater than or equal integer	9000 >= 1000 returns TRUE
>=	3	(Number, Number)	Logical	Greater than or equal floating	10.2 >= 6.4 returns TRUE
>=	3	(Quantity, Quantity)	Logical	Greater than or equal quantity	52 >= 36 returns TRUE
>=	3	(String, String)	Logical	Greater than or equal string (case insensitive)	"ORDER_12" >= "ORDER_22" returns FALSE
>=	3	(Symbol, Symbol)	Logical	Greater than or equal symbol (case insensitive)	"ORDER_12" >= "ORDER_22" returns FALSE
>=	3	(Time, Time)	Logical	Greater than or equal time	021530 >= 010000 returns TRUE
=	3	(Date, Date)	Logical	Equal for date	11MAY1995000000 == 21MAY1995000000 returns FALSE
=	3	(Integer, Integer)	Logical	Equal for integer	9000 == 1000 returns FALSE
=	3	(Number, Number)	Logical	Equal for floating	10.2 == 6.4 returns TRUE
=	3	(Quantity, Quantity)	Logical	Equal for quantity	52 == 36 returns FALSE
==	3	(String, String)	Logical	Equal for string (case insensitive)	"ORDER_12" == "ORDER_22" returns FALSE

Table 1: Operators

Operator	Precedence	Parameters	Return Type	Description	Example
==	3	(Symbol, Symbol)	Logical	Equality for symbol (case insensitive)	"ORDER_12" == "ORDER_22" returns FALSE
==	3	(Time, Time)	Logical	Equality for time	021530 == 010000 returns FALSE
==	3	(Date, Date)	Logical	Inequality for date	11MAY1995000000 != 21MAY1995000000 returns TRUE
!=	3	(Integer, Integer)	Logical	Inequality for integer	9000 != 1000 returns TRUE
!=	3	(Number, Number)	Logical	Inequality for floating	10.2 != 6.4 returns TRUE
!=	3	(Quantity, Quantity)	Logical	Inequality for quantity	52 != 36 returns TRUE
!=	3	(String, String)	Logical	Inequality for string (case insensitive)	"SAW" != "SAW_SAND" returns TRUE
!=	3	(Symbol, Symbol)	Logical	Inequality for symbol (case insensitive)	"SAW" != "SAW_SAND" returns TRUE
!=	3	(Time, Time)	Logical	Inequality for time	021530 != 010000 returns TRUE
%	5	(Integer, Integer)	Integer	Integer remainder	20 % 6 returns 2 (20 / 6 = 3 with a remainder of 2)
%	5	(Number, Number)	Number	Floating remainder	
&	7	(String, String)	String	String concatenation	"ORDER_1" & ".MFG0001" returns "ORDER_1.MFG0001"
and	2	(Logical, Logical)	Logical	Logical AND	If A is TRUE and B is TRUE, then A and B returns TRUE
or	2	(Logical, Logical)	Logical	Logical OR	If A is TRUE and B is FALSE, then A or B returns TRUE
not	1	(Logical)	Logical	Logical NOT	If A is TRUE, then !A returns FALSE

3.2 List Functions

Lists are sequential collections of elements of the same kind.
Here are some examples of lists:

{ "hello", "world" } - a list of two strings
{ 1, 2, 3, 4 } - a list of four integers
{ a, b, c, d, e } - a list of five symbols

It is not possible to mix different kinds of elements into a list:

{ 1, a, "hello" } - illegal list; it must only contains elements of the same kind.

The elements within a list do not need to be unique:

{ 1, 2, 3, 2, 1 } - a list of five integers, with elements 1 and 2 repeated twice.

Lists can be created, queried, filtered, sorted, and manipulated via the functions defined below.

3.2.1 count (List|Void)

count determines how many elements are in a List.

Returns: the number of elements in the List.

Parameters: - a List of any type

Example Usage:

```
count(list(5, 10, 15)); // returns 3
list("a", "b", "c", "d").count; // returns 4
```

3.2.2 filter (List|Void, Expression, Integer)

filter traverses a list copying those elements which match an expression into a new filtered list. The expression can contain the special character '#' which filter substitutes with each element in the list.

Returns: a List of filtered elements.

Parameters: - a List of any type
- an expression which must return Logical

- an optional count of the maximum size for the result. The default is infinite.

Note: filter(list, expr).first automatically sets the last parameter of the filter function to one.

Example Usage:

```
list(2, 4, 6, 8).filter(# < 5); // returns list(2, 4)
list(2, 4, 6, 8).filter(# < 5, 1); // returns list(2)
```

```
filter(list(1, 2, 3), # > 5); // returns an empty list
```

3.2.3 for_each (List|Void, Expression)

for_each traverses a list applying an expression to each element and placing the result in a new List. The expression can contain the special character '#' which filter substitutes with each element in the list.

Returns: a result List with the same number of elements as the given list.

Its elements will have the same type as the expression's return type.

Parameters: - a List of any type
- an expression

Example Usage:

```
list(8, 6, 4, 2).for_each(# / 2); // returns list(4, 3, 2, 1)
list(1, 2, 3).for_each(#).sum; // returns 6
```

3.2.4 sort (List|Void, Expression)

sort orders a list based on an expression. sort processes a list until the sorting expression is true for all adjacent elements. Use 'a' and 'b' as element place holders in the expression.

'sort' uses a fast, non-stable quicksort algorithm. Because it's unstable, you cannot do multiple sorts on multiple keys

(for example: list.sort(a.x > b.x).sort(a.y > b.y)). To do this, use sort_stable for all but the first sort

(for example: list.sort(a.x > b.x).sort_stable(a.y > b.y)) or combine the two sorts into one:

```
{for example: list.sort(if (a.x != b.x), (a.x > b.x), (a.y > b.y)) }
```

Returns: a sorted List of the same type as the given List.

Parameters: - a List of any type
- an sorting expression which must return Logical

Example Usage:

```
list(4, 6, 8, 2).sort(a < b); // returns list(2, 4, 6, 8)
list("l", "h", "m").sort(a > b); // returns list("l", "m", "h")
```


See the documentation for `sort_stable` for an example showing the difference between `sort` and `sort_stable`.

3.2.5

`sort_stable (List(Void, Expression))`

`sort_stable` is just like `sort`, except it uses a slow, but stable sorting algorithm that allows you to cascade sorts. That is, `sort_stable` will ensure that the ordering of the equal elements in the input list is maintained in the output list.

`sort_stable` orders a list based on an expression. `sort` processes a list until the sorting expression is true for all adjacent elements. Use `a` and `b` as element place holders in the expression.

Because `sort_stable` is so much slower than `sort`, use `sort` whenever possible.

Returns: a sorted List of the same type as the given List.

Parameters: - a List of any type
- an sorting expression which must return Logical

Example Usage:

```
list(4, 6, 8, 2).sort_stable(a < b); // returns list(2, 4, 6, 8)
list("i", "h", "m").sort_stable(a > b); // returns list("i", "m", "h")
```

@- operations `sort(a.name < b.name).sort_stable(a.category < b.category)`;

Sorts operations using 'category' as the primary key, and 'name' as the secondary.

The following does the same thing, but is over twice as fast:

`operations.sortIf(a.name != b.name, a.name < b.name, a.category < b.category)`;

@-Example showing the difference between `sort` and `sort_stable`:

```
list(25, 26, 17, 18, 19).sort_stable(integer(a/10) < integer(b/10));
```

This sorts by the most significant digit, and returns `list(17, 18, 19, 25, 26)`

```
@- list(25, 26, 17, 18, 19).sort(integer(a/10) < integer(b/10));
```

This sorts by the most significant digit, and returns `list(19, 17, 18, 26, 25)`

Notice that least-significant digits are in a different order.

3.2.6

`sublist (List(Void, Integer, Integer))`

`sublist` returns a contiguous portion of a list, given a starting element and the desired size of the sublist. If the list does not contain as many elements as requested by the `sublist`, then the size of the resulting sublist is reduced.

Returns: a sublist of the given List; the sublist is the same type as the give List

Parameters: - a List of anything

- the index of an element in the given List;

it becomes the first element of the resulting sublist;

must be > 0

- the number of elements in the resulting sublist;

must be > 0

For example:

```
list(1, 2, 3, 4, 5).sublist(2, 3); // returns list(2, 3, 4)
list("a", "b", "c").sublist(3, 10); // returns list("c")
sublist(list(-1, 2, 0), 0, 0); // returns list("")
list(1, 2, 3).sublist(0, 2); // returns an error
list(1, 2, 3).sublist(1, 0); // returns list("")
list(1, 2, 3).sublist(5, 2); // returns list("")
```

3.2.7

`unique (List(Void))`

`unique` removes duplicate elements from a list.

WARNING: `unique` only works for data-types whose representation can be contained in a single word, so it does not work for types such as `String`, `Period`, `Quantity`, `Time`, and `Unit`. Since it is difficult for users to know which types will work, and which will not, try it and see. This limitation will be fixed someday.

Returns: a List of unique elements

Parameters: - a List of single word data types

Example Usage:

```
unique(list(1, 2, 3, 2, 1)); // returns list(1, 2, 3)
```

3.2.8 contains (List(Void), Void)

contains determines if a list contains a certain element. It returns 'true' only if the 2nd parameter is an exact match for one of the elements in the list given as the first parameter.

Parameters:

- a List of any type
- the element to look for

Example Usage:

```
define(buffer1, sites.find("site1").buffers.find("one"));
define(buffer2, sites.find("site2").buffers.find("one"));
sites.find("site1").buffers.contains(buffer1); // Returns TRUE
sites.find("site1").buffers.contains(buffer2); // Returns FALSE
// It returns FALSE, because even though the key-fields for buffer1 and
// buffer2 are the same, they're different objects. An exact match must be found.
sites.find("site1").buffers.find(buffer2.name); // Returns TRUE
```

```
@_ list(1, 2, 3, 4).contains(0); // returns False
contains(list(3, 5, 7, 9), 7); // returns True
```

3.2.9 found (List(Void), Void)

Given a list of models, the search uses the Model's 'key_field', but only returns TRUE if the matching key-field model is found. It's just like 'find', only 'found' one returns a Logical instead of the element. For example, 'found(key)' works just like 'find(key)', except it doesn't generate an error message when the key isn't found.

Returns: True when the given list contains an element with the same key-field;

False otherwise.

Parameters:

- a List of any type
- the element or 'key_field' to look for (the 'key_field' must be the same as the Model's 'key_field' type)

Example Usage:

```
supply_chains.found("test"); // returns True if there is a supply
// chain named "test" in the supply
// chain list
```

3.2.10 element (List(Void), Integer)

element returns the requested element from a list. Passing 1 in as the second argument returns the first element in the list.

Returns: The nth element of a List. The returned element retains its type.

element reports an error if there is no such element.

Parameters:

- a List of any type
- requested element (an Integer > 0)

Example Usage:

```
element(list(5, 10, 15), 1); // returns 5
list("a", "b", "c").element(2); // returns "b"
list(1, 2, 3, 4).element(5); // reports an error
```

3.2.11 find (List(Void), Void)

find searches a list of models and returns the first model whose 'key_field' matches a value. This version of find is used when the first argument is a list of models.

Returns: a Model with 'key_field' matching the given value.

The model's type is retained.

find reports an error if no match is found.

Parameters:

- a Model List
- desired 'key_field' value (must be same type as the Model's 'key_field' type).

Example Usage:

```
supply_chains.find("test"); // returns the supply chain named "test"
site.find("Dallas"); // returns the site named "Dallas"
```

3.2.12 find_or_nonexistent (List(Void), Void)

find searches a list of models and returns the first model whose 'key_field' matches a value. This version of find is used when the first argument is a list of models. This is exactly like 'find' except 'nonexistent' is returned when no match is found, instead of reporting an error.

Returns: a Model with 'key_field' matching the given value.

The model's type is retained.

Parameters: - a Model List

- desired `key_field` value (must be same type as the Model's `key_field` type).

Example Usage:

```
supply_chains.find_or_nonexistent("test"); // returns nonexistent  
site.find_or_nonexistent("Dallas"); // returns the site named "Dallas"
```

3.2.13 find_or_create (List|Void, Void)

`find_or_create` searches a list of models and either:

- * returns the first model whose `key_field` matches a value, or
- * if the model does not exist, creates and appends it to the end of the list followed by returning the newly created model.

`find_or_create` is only available to a list of models.

Returns: a Model with `key_field` matching the given value.

The model's type is retained.

Parameters: - a Model List,

- desired `key_field` value (must be same type as the Model's `key_field` type)

Example Usage:

In the following expressions, assume that the supply chain named "test" does not exist. Observe that the first expression yields an error because it did not find "test". The other expressions return the model named "test".

```
supply_chains.find("test");  
supply_chains.find_or_create("test");  
supply_chains.find("test");  
supply_chains.find_or_create("test");
```

3.2.14 list_index (List|Void, Void)

`list_index` searches a list of models and returns the position in the list of the first model whose `key_field` matches a value. It also searches an arbitrary list for a match (using the `==` operator). `list_index` reports an error if the list element is not found.

Returns: `some_list[some_list.list_index(key)]` is equivalent to `some_list.find(key)`

Parameters: - a List

- desired `key_field` value (must be same type as the Model's `key_field` type), or an element of the list.

Example Usage:

```
list("a", "b", "c").list_index("b") // returns 2
```

3.2.15 first (List|Void)

`first` returns the first element of a list.

Returns: The first element of the List. The returned element retains its type.

@-first returns NONEXISTENT if the List is empty.

Parameters: - a List of any type

Example Usage:

```
first(list(5, 10, 15)); // returns 5  
list("one", "two", "three").first; // returns "one"
```

3.2.16 last (List|Void)

`last` returns the last element of a list.

Returns: The last element of the List. The returned element retains its type.

`last` returns NONEXISTENT if the List is empty.

Parameters: - a List of any type

Example Usage:

```
last(list(5, 10, 15)); // returns 15  
list("one", "two", "three").last; // returns "three"
```

3.2.17 list ()

`list` creates a new list of elements.

Returns: a List containing the passed in elements, in the order passed in.

Parameters: - variable: any number of elements of the same type
Any type is valid. At least one element must be provided. Nonexistent parameters are ignored.

@- An argument can itself be a list. However, list() will not create a list of lists. If an argument is a List, then its elements will be extracted and placed into the created list.

Example Usage:

```
list(3, list(2, 4, 6), 1); // returns a List of 5 Integers
// where "3" is the first element,
// "4" is the third element, and
// "1" is the fifth element.
list(symbol(a), symbol(b)); // returns a List of 2 symbols.
list(3, "three"); // returns an error since the given
// elements are different types
// (Integer and String)
list(): // returns an error (there must be at
list(make_type(nonexistent, string)); // returns an empty list[string]
```

3.2.18 integers (Integer, Integer)

Generate a list of integers from min to max. This is used as a general looping mechanism when combined with 'for_each'.

Example:

```
integers(1, 5) -> integers(1, 5) = 1, 2, 3, 4, 5
integers(5, 4) -> integers(5, 4) =
integers(5, 5) -> integers(5, 5) = 5
integers(5, 6) -> integers(5, 6) = 5, 6
```

Returns: A list of integers

Parameters: - min, max

Example Usage: integers(1, 3).for_each(echo(#));

3.3 Multi_Keyed_List Functions

Multi_Keyed_Lists are multi-hash-keyed collections of multiple elements of the same kind.

There are actually two types of multi-keyed-lists:

- 1) Those created by multi_key_bucketize (see the multi_key_bucketize function for more detail)
- 2) Those created by multi_keyed_list (see the multi_keyed_list function for more detail)

An Multi_Keyed_Lists can be created, and queried via the functions in its group. Note: they may also be used by any of the generic List functions as well. (e.g. element, for_each, num, etc.)

3.3.1

key (Symbol, Symbol, Logical, Logical)
@-Returns: a key definition for use by multi_key

Parameters:

- name: the name of the key
- extract_expr: an expression (usu. containing #) that returns a key value symbol
- filter_expr: an expression (usu. containing #) that returns a logical indicating whether this key value will be excluded
- active: a logical that indicates whether this key is active or not

Example Usage:

```
define(key1, key("item", #.name, #.name != "leg", TRUE);
```

3.3.2

multi_key (Symbol, Void)
@-Returns: a list of key definitions for use by multi_keyed_list and multi_key_bucketize

Parameters:

- type: the name of the type that this key definition will be used for, and one or more of the following
- key: a key definition (the output of the OIL function 'key')

Example Usage:

```
define(mk, multi_key("Item",
    key1, // this is from the 'key' example
    key("where", #.ownername, #.ownername != "CANADA", TRUE));
```

3.3.3 multi_keyed_list (List(Void), List(Void), Expression)

@-Returns: a multiply-keyed list of lists of elements

Parameters:

- values: a List of any type, for processing
- keys: a List of hash-key definitions (from multi_key) and one or more of the following (but all of the same type)
- element: an expression that returns any type, evaluated for each element of values

Example Usage:

```
define(mkl, item_list, multi_keyed_list(mk, // this is from the 'multi_key' example
#name,
#category,
#delivery name));
```

3.3.4 multi_keyed_element (List(Void), Integer, Symbol, Symbol)

@-Returns: a list of the element(s) stored at the specified location(s)

Parameters:

- values: a List of any type, but created only by multi_keyed_list
- integer: the specific element in the list at that hash-keyed location to retrieve and one or more pairs of the following
- symbol: a key name.
- symbol: a key value.

Notes: - The number of specified key name-value pairs need not be as many as were used to build the original multi_keyed_list. Any keys not specified will be treated as if they were specified, but with a wildcard.

- Key name order does not need to be in the same order as it was when the list was built

Example Usage:

Here we get the list of all the #category that were stored for all site.names == "USA"

```
mkl_multi_keyed_element(2,                // mkl is from the multi_keyed_list example
```

"where", "USA");

3.3.5 key_names (List(Void))

Use key_names to get at the list of key names used to build this multi_keyed_list or multi_keyed_bucketized list.

Returns: a list of Symbols

Parameters:

- values: a Multi Keyed List of any type

Example Usage:

```
// mkl is from the multi_keyed_list example
mkl.key_names() = "item", "where"
```

3.3.6 key_values (List(Void), Symbol)

Use key_values to get at the list of unique key values for the given name used to build this multi_keyed_list or multi_keyed_bucketized list.

Returns: a list of Symbols

Parameters:

- values: a Multi Keyed List of any type
- name: the name of a key used to create the list

Example Usage:

```
// mkl is from the multi_keyed_list example
mkl.key_values("item") = "table", "chair", "seat"
```

3.3.7 multi_key_bucketize (List(Void), List(Void), List(Date_Range), Expression, Expression)

Returns: a multiply-keyed and date_ranged (bucketized) list of lists of elements

Parameters:

- values: a List of any type, for processing
- keys: a List of hash-key definitions (from multi_key)
- date_ranges: a List of contiguous date ranges (defines the buckets)
- date: an expression (usu. containing #) that returns a date, evaluated for each element of values
- element: an expression (usu. containing #) that returns any type, evaluated for each element of values

Example Usage: (see bucketize and multi_keyed_list)

3.3.8 multi_key_bucketize_element (List(Void), Integer, Date, Symbol, Symbol)
@-Returns: a list of the element(s) stored at the specified location(s)

Parameters:

- values: a List of any type, but created only by multi_key_bucketize
- integer: the specific element in the list at that hash-keyed location to retrieve
- date: an expression that returns a Date.
- and one or more pairs of the following
- symbol: a key name.
- symbol: a key value.

Notes: - The number of specified key name-value pairs need not be as many as were used to build the list in the originating call to multi_key_bucketize. Any keys not specified will be treated as if they were specified, but with a wildcard.

- Key name order does not need to be in the same order as it was when the list was built

Example Usage: (see bucketize and multi_keyed_list)

3.4 Singly_Keyed_List Functions

@-Singly_Keyed_Lists are hash-keyed collections of elements of the same kind.

There are actually two types of singly-keyed-list:

- 1) Those created by bucketize (see the bucketize function for more detail)
- 2) Those created by keyed_list (see the keyed_list function for more detail)

An Singly_Keyed_List can be created and queried via the functions in its group.

Note: they may also be used by any of the generic List functions as well.
(e.g. element, for_each, num, etc.)

3.4.1

bucketize (List(Void), List(Date, Range), Expression, Expression)

Use bucketize when you want to be able to quickly retrieve values from a large list that is date-based in nature. It organizes the input list into a table with one column per date-range (from the 2nd parameter), and any number of rows(indexed by a symbol). The intersection (table entries) are a list which you can access using the 'bucket_list' function. The last two parameters are used to determine which of the input items go into which symbol/date_range intersection list.

Returns: a list of values, organized for very quick retrieval characteristics.

Parameters:

- values: a List of any type
- date_ranges: a List of Date_Range
- date expression: an expression that returns a Date.
- symbol expression: an expression that returns a Symbol.

Description:

- 1) Takes the list of Date_Range, and makes a bucket for each one (Note that in a list of Date_Range buckets, that any two adjacent buckets, A and B, must have: A.end == B.start)
 - 2) Then it loops over the list of values and does the following for each:
 - 2.1) evaluates the date expression
 - 2.2) finds the bucket that the date expression fits in (i.e. bucket.start <= date < bucket.end) (Note this implies that given a bucket where bucket.start == bucket.end, nothing can ever fall within it)
 - 2.3) evaluates the symbol expression
 - 2.4) puts this value indexed by this symbol to the list in this bucket (NOTE: if more than one value is stored with the same key-symbol, then all values with that key-symbol will be returned by the single call to bucket_list for that key-symbol.)
- Example Usage: take a list of Item_Promises and put each one in the bucket whose date range contains the given date as evaluated in the expression #.owner.date.start, and hash-keys it with the given Symbol as evaluated in the expression #.item.name.

Basic Functions	bucket_list (List Void, Date, Symbol)
-----------------	---------------------------------------

```
variable ips = delivery_promises.for_each(#item, promises);
variable bi = bucketize(date_range_list,
    #ownerdue_start,
    #item.name);
```

Extended first time user example: Contrived and probably completely useless non-model example.
but it shows in painstaking detail the flow of data:

```
defnecdate_range_list, list(date_range("95/01/01 00:00 / 95/01/02 00:00"), //
    bucket#1
    date_range("95/01/02 00:00 / 95/01/06 00:00"), // bucket#2
    date_range("95/01/06 00:00 / 95/01/10 00:00"), // bucket#3
    date_range("95/01/10 00:00 / 95/01/14 00:00"), // bucket#4
    date_range("95/01/14 00:00 / 95/01/20 00:00")) // bucket#5

defnecb_example, integers(5,10), bucketize(date_range_list,
    date("95/01/00 00:00") + time(#string & " day"),
    #string))
```

So for each of the integer numbers between 5 and 10, we generate a date for it, (Specifically: "95/01/05 00:00", "95/01/06 00:00", "95/01/07 00:00", "95/01/08 00:00", "95/01/09 00:00", "95/01/10 00:00")

And therefore, the integer 5's value gets stored in bucket#2 since:
bucket#2.start <= (5's date) < bucket#2.end
"95/01/02 00:00" <= "95/01/05 00:00" < "95/01/06 00:00"
Integer 6 ends up in bucket#3, (NOT in bucket#2, due to the non-inclusive nature of the end date of a bucket's defining date-range)
Integer 7, 8 and 9 also get put in the list in bucket#3 as well.
Integer 10 ends up in bucket#4 (NOT in bucket#3, same reason as integer 6)

3.4.2

bucket_list (List|Void, Date, Symbol)
Use bucket_list to quickly retrieve a specific value (or values) from a list created by bucketize.

Returns: a list of values

Basic Functions	bucket_list_by_date (List Void, Date)
-----------------	---------------------------------------

Parameters:

- values: a List of any type (but created by bucketize)
- date: a Date.
- Symbol: a Symbol.

Description:

1) Finds the bucket that the given Date falls within its Date Range (i.e. bucket.start <= date < bucket.end) 2) Retrieves any values that were stored in that bucket with a lookup key that matches the given Symbol.

Example Usage: (see bucketize for definition of b_example)

```
b_example.bucket_list(date("95/01/10 00:00"), "9") --> nothing
b_example.bucket_list(date("95/01/02 20:00"), "9") --> nothing
NOTE: These examples also show how the given lookup date does not have to match the original date used to put the data in that specific bucket:
b_example.bucket_list(date("95/01/02 20:00"), "5") --> 5
b_example.bucket_list(date("95/01/07 00:00"), "8") --> 8
b_example.bucket_list(date("95/01/06 00:00"), "8") --> 8
b_example.bucket_list(date("95/01/06 00:00"), "9") --> 9
b_example.bucket_list(date("95/01/06 00:00"), "adam") --> nothing
b_example.bucket_list(date("95/01/09 23:59"), "8") --> 8
```

3.4.3

bucket_list_by_date (List|Void, Date)

Use bucket_list_by_date to quickly retrieve the whole list of values in the date_range bucket that the given date falls within.
Returns: a list of values organized by key (i.e. the same kind of list as comes out of keyed_list, and therefore usable by keyed_element, and keys)

Parameters:

- values: a List of any type (but created by bucketize)
- date: a Date.

Description:

- 1) Finds the bucket that the given Date falls within its Date_Range (i.e. bucket.start <= date < bucket.end)
- 2) Returns all the values that were stored in that bucket.

Example Usage: (see bucketize for definition of b_example)

b_example.bucket_list_by_date(date("95/01/06 00:00")) --> 6, 7, 8, 9

3.4.4 bucket_list_by_key (List(Void), Symbol)

Use bucket_list_by_key to quickly retrieve the whole list of values in all the date_range buckets that match the given key.

Returns: a list of values organized by key (i.e. the same kind of list as comes out of keyed_list, and therefore usable by keyed_element, and keys)

Parameters:

- values: a List of any type (but created by bucketize)
- symbol: a Symbol/String key.

Description:

- 1) Searches for the given key in every bucket in the list.
- 2) Returns the values found in a Keyed_List that is keyed by the date_range.string of the bucket it was found in.

@-Example Usage: (see bucketize for definition of b_example)

b_example.bucket_list_by_key("8").keys() --> 95-01-12 00:00 / 95-01-20 00:00

Note: the default delimiter for date-ranges is a '-', not a 'y', hence the above output. What the user should also derive from this is that given the two following expressions that seemingly should give the same output, only the first will give any output, due to the default date delimiter as mentioned above.

b_example.bucket_list_by_key("8").
keyed_element(date_range("95/01/06 00:00 / 95/01/10 00:00").string) --> 8
b_example.bucket_list_by_key("8").
keyed_element("95/01/06 00:00 / 95/01/10 00:00") --> NOTHING

If you must do it that way, this will work:
b_example.bucket_list_by_key("8").
keyed_element("95-01-06 00:00 / 95-01-10 00:00") --> 8
However it is recommended to use date_range().string to keep away from those types of subtleties.

3.4.5 bucket_symbols (List(Void))

Use bucket_symbols to get at the list of unique Symbols (keys) that were used in building the given list during bucketize.

Returns: a list of Symbols

Parameters:

- values: a List of any type (but created by bucketize)

Example Usage: (see bucketize for definition of b_example)
b_example.bucket_symbols() --> "5", "6", "7", "8", "9", "10"

3.4.6 keyed_list (List(Void), Expression)

Use keyed_list when you want to be able to quickly retrieve values from a large list

Returns: a list of values, organized for very quick retrieval characteristics.

Parameters:

- values: a List of any type
- symbol expression: an expression that returns a Symbol.

Description:

- 1) It loops over the list of values and does the following for each:

- 1.1) evaluates the symbol expression
- 1.2) stores this value, indexed by this symbol, in the new list. (NOTE: if more than one value is stored with the same key-symbol, then all values with that key-symbol will be returned by the single call to keyed_element for that key-symbol.)

Example Usage:

Basic Functions

keyed_element (Symbol)

```
define(keyed, integers(1..10),keyed_list(#string & "0"))
```

This one chooses to store the integer values:

```
1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

with the respective key-symbols of:

```
"10", "20", "30", "40", "50", "60", "70", "80", "90", "100"
```

Use `keyed_element` to retrieve a value from a `keyed_list`:
`keyed_element("50").sum -> 5`

3.4.7

keyed_element (Symbol)

Use `keyed_element` to quickly retrieve a specific value (or values) from a list created by `keyed_list`.

Returns: a list of values

Parameters:

- values: a List of any type (but created by `keyed_list`)

- Symbol: a Symbol

Description:

Retrieves any values that were stored in this list with a lookup key that matches the given Symbol. Retrieval is not done by a traditional linear-search, therefore as the lists get larger, the lookup speed of this method improves handsomely over linear-search.

Example Usage:

```
define(keyed, integers(1..10),keyed_list(#string & "0"))
```

```
keyed_keyed_element("50") -> list(5)
```

```
keyed_keyed_element("5") -> nonexistent
```

3.4.8

keys (List(Void))

Use `keys` to get at the list of unique Symbols (keys) that were used in building the given list during `keyed_list`.

Returns: a list of Symbols

Parameters:

Basic Functions

Recurse_List Functions

- values: a List of any type (but created by `keyed_list`)

Example Usage:

```
define(keyed, integers(1..10),keyed_list(#string & "0"))
```

```
keyed_keys() -> "10", "20", "30", "40", "50", "60", "70", "80", "90", "100"
```

3.5 Recurse_List Functions

`Recurse_List` are lists of elements created by `recurse` or `recurse_or_trim` and have the added feature of storing the recurse depth information. This can be handy for nesting/indentation.

Note: they may also be used by any of the generic List functions as well.
(e.g. `element`, `for_each`, `num`, etc.)

3.5.1

recurse (List(Void), Expression)

`recurse` uses an expression to recursively process each element in a list. The recursion ends when the expression returns an empty list.

`recurse` starts by placing each element in a list in a result list. Each element of the result list is then passed to the expression where it is substituted for the special placeholder `#`. This continues until the expression returns an empty list.

While executing 'expression', the variable 'parents' is bound to the list of all the ancestors of `#`. The most recent ancestor is `parents.first`, the root of the recursion is `parents.last`.

Returns: a List of the same type as the given list

Parameters: - a List of any type

- an expression which must return List's type

Example Usage:

Consider a `List(Number)` containing two Numbers { 12, 20 } and an Expression that returns the list of the factors of a Number, excluding itself. Calling `recurse` on those will result in a `List(Number)`. The first Number in the resultant List will be "12". The Expression is evaluated with `#` set to "12" resulting in the List { 2 3 4 6 }. Then `recurse` evaluates itself with the same Expression parameter and the newly generated List. So, the next Number in the output List is "2" { 12 2 ... }. Evaluating the Expression with `#` set to "2" results in an empty List, so the recursion stops. Thus, the next

element in the output List is "3" { 12 2 3 ...}. The Expression also returns empty List with "3", so the next element is "4" { 12 2 3 4 ...}. Evaluating the Expression with '#' set to "4" results in the List { 2 2 }. So, the next element in the result is "2" { 12 2 3 4 2 ...}. This continues until the final List { 12 2 3 4 2 2 6 2 3 20 2 4 2 2 5 10 2 5 } results.

recurse is particularly useful in the cell of a replicating Worksheet. In that context it has an additional effect: the depth of the recursion of each element is known. Various Layouts and Controls can take advantage of that information to let you see the "nesting", to let you see that the "2" and the "3" are under/within the "6", which, along with the "2", "3", and "4", is under/within the "12". For example, the *indent_general* Control will indent proportional to the depth of the recurse that computed the value.

The best example of the use of recurse and *indent_general* Control is the generation of an "indented bill-of-material". A function that gives the List of components of an Item gives a "single-level bill-of-material (BOM)". The recurse function allows the same function to be evaluated for each of those components to get the List of their components. And so on until the Items with no components are reached. Thus, recurse with the "single-level BOM" function generates a "multi-level BOM". By putting these in a column of a replicating Worksheet using a *indent_general* Control, the Items will be indented according to their depth in the BOM. The result is an "indented BOM".

Limitations:

The result from 'recurse' is a very special list, with the indent information attached. If the list gets copied by 'filter' or 'for_each' (or anything else) the indent info gets lost, which means the *indent_general* control will not indent. The work-around is to do all your filtering *inside* the recurse.

To avoid infinite recursions, 'recurse' is limited to a recursion depth of 1000, and it will not generate a list larger than 10,000 elements. You can change these limits with the 'recurse_depth' and 'recurse_items' options.

3.5.2 recurse_and_trim (List|Void, Expression, Expression)

recurse_and_trim is like the 'recurse' function, with an additional Logicalian "filter" expression which is used to trim the resulting tree.

The 'filter' expression argument is a little different from the normal "filter" OIL function, because an element for which the expression returns "false" will still be included if one of it's children's filter expressions returned "true". In other words, this filter is specifically designed to trim dead branches of the tree.

While executing both the recursion expression and the filter expression, the variable 'parents' is bound to the list of all the ancestors of '#'. The most recent ancestor is parents.first, the root of the recursion is parents.last.

Returns: a List of the same type as the given list

Parameters: - a List of any type
- an expression which must return List's type
- an expression which returns a logical

3.5.3 current_depth (Cell)

Returns the current depth of a (recursively) replicating cell

Returns: An integer greater than or equal to zero

Parameters: - A cell name or id

Example Usage:

current_depth(C1):

3.6 Text_String Functions

Text strings are composed of zero or more characters. Text strings can be directly specified using pairs of double quotes. Nonprintable control characters can be specified by preceding them with a backslash (\). For instance, control-A is specified as "\a", control-B as "\b", etc. Here are some sample strings:

```
@-"Dallas"
"The quick brown fox jumped over the fence."
"$%#@#"
"" (an empty string)
"\\" (a string containing a single double quote)
"\n" (a string containing a single backslash)
"Hello\World!:" (a string which embeds two control characters - a new line and a tab)
```

Strings can be created, edited, compared, and searched using the functions defined below.

3.6.1

string (Void)
string converts anything to a string, using the default format. The default format used depends on the argument's type.

Returns: a string

Parameters: - any type

Example Usage:

```
string(3,14); // returns 3,14
string(quantity(100)); // returns 100
string(logical("yes?")); // returns Yes
string("test"); // returns test
string(list(1, 2, 3)); // returns 1, 2, 3
```

3.6.2 string (Void, Symbol)

string converts anything to a string, using the named format. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: a string

Parameters: - any type

- the named format (a Symbol) to use

Example Usage:

3.6.3 index (String, String)

index returns the character index of the first occurrence of 'sub-string' in 'string'. If 'sub-string' is not found, the length of 'string' plus 1 is returned (the index of the character that is one past the end of the string). The character index is one based (i.e., the first character of a string is at index one).

Equivalent to `index(string, sub-string, 1)`;

Returns: an integer

Parameters: - the String to search

- the sub-String to find

Example Usage:

```
index("one - two - three", "on"); // returns "1"
index("one - two - three", "-"); // returns "6"
index("sub-string not found", "test"); // returns "21"
```

3.6.4 index (String, String, Integer)

index returns the character index of the nth occurrence of 'sub-string' in 'string'. If 'sub-string' is not found, the length of 'string' plus 1 is returned (the index of the character that is one past the end of the string). The character index is one based (i.e., the first character of a string is at index one).

Returns: a String which contains either a character index or a string length

Parameters: - the String to search

- the sub-String to find

- the nth occurrence (an Integer); must be > 0

Example Usage:

```
index("one - ion - done", "on", 3); // returns "14"
index("one - two - three", "- ", 2); // returns "12"
index("sub-string not found", "test", 1); // returns "21"
```

3.6.5 left (String)

left returns the first character of a string.

Equivalent to `left(string, 1)`;

Returns: the first character of a string

Parameters: - the original String

Example Usage:

```
left("planner"); // returns "p"
```

3.6.6 left (String, Integer)

left returns the first 'n' characters of a string. If 'n' is greater than the number of characters in the string, left returns the original string.

Returns: the first 'n' characters of a string

Parameters: - the original String

- the number of characters to return; must be >= 0

Example Usage:

Basic Functions	left (String, String)
-----------------	-----------------------

```
left("planner", 4); // returns "plan"
left("abcde", 6); // returns "abcde"
left("test", 0); // returns ""
```

3.6.7 left (String, String)

left uses delimiters to split a string and returns the first, or leftmost, sub-string. If no delimiters are found, the original string is returned.

Equivalent to mid(string, delimiters, 1);

Returns: the leftmost sub-String or the original String

Parameters: - the original String
- the delimiters (a String)

Example Usage:

```
left("one - two - three", "-"); // returns "one"
left("delimiter not found", "x"); // returns "delimiter not found"
```

3.6.8 lower (String)

lower converts all uppercase characters in a string to lowercase.

Returns: a String containing only lowercase characters

Parameters: - a String

Example Usage:

```
lower("Plan #35"); // returns "plan #35"
```

3.6.9 mid (String, Integer)

mid returns the 'nth' character of a string. If the requested character is beyond the end of the string, mid returns the last character of the string.

Equivalent to mid(string, n, 1);

Returns: the requested character of a string or the last character of the string

Parameters: - the original String
- the requested character (an Integer); must be > 0

Basic Functions	mid (String, Integer, Integer)
-----------------	--------------------------------

```
Example Usage:
mid("planning", 3); // returns "a"
mid("test", 6); // returns ""
```

3.6.10 mid (String, Integer, Integer)

mid returns the middle 'k' characters starting at the 'nth' character of a string. If the starting character is beyond the end of the string, mid returns an empty string. If the requested number of characters run beyond the end of the string, mid returns just the characters from the starting character to the end of the string.

Returns: the requested sub-String, or portion thereof, or an empty string

Parameters: - the original String
- starting character (an Integer); must be > 0
- the number of characters to return; must be >= 0

Example Usage:

```
mid("planner", 2, 3); // returns "an"
mid("abcde", 4, 5); // returns "de"
mid("test", 6, 2); // returns ""
mid("test", 3, 0); // returns ""
```

3.6.11 mid (String, String, Integer)

mid uses delimiters to split a string and returns the requested sub-string. If no delimiters are found, mid returns an empty string.

Returns: the requested sub-String or an empty String

Parameters: - the original String
- the delimiters (a String)
- the requested sub-String (an Integer); must be > 0

Example Usage:

```
mid("one - two - three", "- ", 2); // returns "two"
mid("delimiter not found", "x", 3); // returns ""
```

3.6.12 proper (String)

proper capitalizes the first character of every word in a string, and converts the remain characters to lowercase. Specifically, proper capitalizes any letter not preceded by a letter. Otherwise, the letter is converted to lowercase. Underscores are replaced with spaces.

Returns: a proper String

Parameters: - the original String

Example Usage:

```
proper("Truly_INTEGRATED planning"); // returns "Truly Integrated Planning"
proper("this is 'hard to read'"); // returns "This is 'Hard to Read'".
```

3.6.13 justify (String, Integer, String)

The result is the first parameter with newlines inserted such that the width between newlines will be less than or equal to width (the 2nd parameter). Lines may be broken at break_on characters (the 3rd parameter).

Returns: a string

Parameters: - The original String.

- The desired maximum width.
- The set of characters on which to break the string.

Example Usage:

```
some_string.justify(40, ".,:");
```

3.6.14 replace (String, Integer, Integer, String)

replace performs sub-string substitution. The replacement string can be any length (including the empty string). replace inserts text if the number of characters to replace is zero. replace deletes text if the replacement string is empty. replace appends text if the starting location is beyond the end of the string (number of characters is irrelevant). replace appends text to the beginning if the starting location is 1 and the number of characters is 0.

Returns: a String with the requested substitution

Parameters: - the original String

- the starting location (an Integer); must be > 0
- the number of characters to replace (an Integer); must be >= 0
- the replacement String

Example Usage:

```
replace("a master plan", 3, 6, "global"); // returns "a global plan"
```

```
replace("planning", 2, 4, ""); // returns "ping"
replace("planning", 6, 0, "er k"); // returns "planner king"
replace("plan", 6, 2, "ning"); // returns "planning"
replace("plan", 1, 0, "master"); // returns "master plan"
```

3.6.15 right (String, String)

right uses delimiters to split a string and returns the last, or rightmost, sub-string. If no delimiters are found, a copy of the original string is returned.

Equivalent to mid(string, delimiters, count(delimiters)).

Returns: the rightmost sub-String or a copy of the original String

Parameters: - the original String

- the delimiters (a String)

Example Usage:

```
right("one - two - three", "-"); // returns "three"
right("delimiter not found", "x"); // returns "delimiter not found"
```

3.6.16 right (String)

right returns the last character of a string.

Equivalent to right(string, 1);

Returns: the last character of a string

Parameters: - the original String

Example Usage:

```
left("planer"); // returns "r"
```

3.6.17 right (String, Integer)

right returns the last 'n' characters of a string. If 'n' is greater than the number of characters in the string, right returns the original string.

Returns: the last 'n' characters of a string

Parameters: - the original String

- the number of characters to return; must be >= 0

Example Usage:
exp(0); // returns 1.0
exp(-1); // returns 0.367879
exp(2.71); // returns 15.0293

3.7.9 integer (Number)

integer converts a number to an integer, ignoring the number's fractional part.

Returns: the Integer portion of the given Number

Parameters: - a Number

Example Usage:
integer(4.2); // returns 4
integer(0.9); // returns 1
integer(-4.6); // return -4

3.7.10 integer (Percentage)

integer converts a percentage to an integer. The percentage is truncated to the nearest integer.

Returns: the Integer representation of the Percentage

Parameters: - a Percentage

Example Usage:
integer(percentage("25%")); // returns 0
integer(percentage("28.5%")); // returns 2
integer(percentage("99.5%")); // returns 99
integer(percentage(0.25)); // returns 0

3.7.11 integer (String)

integer converts a string to an integer using the default format.

Returns: the Integer representation of the String

Parameters: - a String; must only contain digits and/or a negative sign

Example Usage:
number("22"); // returns 22

number("-100"); // returns -100
number(""); // returns 0

3.7.12 integer (String, Symbol)

integer converts a string to an integer using the named format. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: the Integer representation of the formatted String

Parameters: - a String
- the named format (a Symbol) to use

Example Usage:

3.7.13 ln (Number)

ln computes the natural (base e) logarithm of a number.

Returns: the natural logarithm of a Number

Parameters: - a Number; must be >= zero

Example Usage:
ln(0.0); // returns -INFINITE
ln(2.71); // returns 0.996949
ln(-1.0); // returns an error

3.7.14 log (Number)

log computes the base-10 logarithm of a number. Note that log(x) is the same as log(x, 10) and log10(x).

Returns: the base-10 logarithm of a Number

Parameters: - a Number; must be >= zero

Example Usage:
log(0.0); // returns -INFINITE
log(10.0); // returns 1.0
log(-1.0); // returns an error

3.7.15 log (Number, Number)

log computes the logarithm of a number to the specified base.

Basic Functions	log10 (Number)
-----------------	------------------

Returns: the logarithm of a Number using the specified base

Parameters: - a Number; must be >= zero
- a base (Number); must be > zero and != to 1

Example Usage:

```
log(0.0, 0.1); // returns -INFINITE
log(10.0, 5.0); // returns 1.43068
log(-1.0, 1.0); // returns an error
log(1.0, -1.0); // returns an error
```

3.7.16 log10 (Number)

log10 computes the base-10 logarithm of a number. Note that log10(x) is the same as log(x, 10) and log(x).

Returns: the base-10 logarithm of a Number

Parameters: - a Number; must be >= zero

Example Usage:

```
log(0.0); // returns -INFINITE
log(10.0); // returns 1.0
log(-1.0); // returns an error
```

3.7.17 measure (Quantity)

measure converts a quantity to a measure.

Returns: the Measure category of the given quantity.

Parameters: - a Quantity

Example Usage:

```
measure(quantity("32")); // returns unitless
measure(quantity("1 kg")); // returns mass
measure(quantity("2 min")); // returns time
```

3.7.18 measure (String)

measure converts a string to a measure using the default format. The string must be a defined measure.

Basic Functions	measure (String, Symbol)
-----------------	----------------------------

Returns: the Measure representation of the String or an error if not a defined measure

Parameters: - a String; must be a defined measure

Example Usage:

```
measure("length"); // returns length
measure("time"); // returns time
measure(""); // returns unitless
measure("any"); // returns an error
measure("infinite"); // returns an error
```

3.7.19 measure (String, Symbol)

measure converts a string to a measure using the named format. The string must be a defined measure. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: the Measure representation of the String or an error if not a defined measure

Parameters: - a String; must be a defined measure
- the named format (a Symbol) to use

Example Usage:

3.7.20 money (String)

money converts a string to a money quantity using the default format. It is important to note that the money quantity can represent any specific national denomination, such as US dollars or Japanese yen.

Returns: the money Quantity representation of the String

Parameters: - a String; the string must contain a numeric value

Example Usage:

```
money("0"); // returns 0.00
money("10.00"); // returns 10.00
money("6.66666"); // returns 6.67
money("$5"); // returns an error
```

3.7.21 money (String, Symbol)

money converts a string to a money quantity using the named format. It is important to note that the money quantity can represent any specific national denomination, such as US dollars or Japanese yen. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: the money Quantity representation of the String in the named format

Parameters: - a String; the string must contain a numeric value
- the named format (a Symbol) to use

Example Usage:

3.7.22 number (Integer)

number converts an integer to a number.

Returns: the Number representation of the given integer

Parameters: - an Integer

Example Usage:

```
number(0); // returns 0.0  
number(-1); // returns -1.0  
number(1000000); // returns 1000000.0
```

3.7.23 number (Percentage)

number converts a percentage to a number using the default format.

Returns: the Number representation of the given percentage.

Parameters: - a Percentage

Example Usage:

```
number(percentage("20%")); // returns 0.20  
number(percentage("33%")); // returns 0.33  
number(percentage("150%")); // returns 1.50
```

3.7.24 number (Quantity)

number converts a unitless quantity to a number.

Returns: the Number representation of the given unitless quantity.

Parameters: - a Quantity; must be unitless

Example Usage:

```
number(quantity("32")); // returns 32.0  
number(quantity("-1")); // returns -1.0
```

3.7.25 number (String)

number converts a string to a number using the default format. Strings "infinite" and "-infinite" are converted to positive and negative infinity respectively.

Returns: the Number representation of the String

Parameters: - a String; must only contain digits and/or a single decimal point and/or a negative sign

Example Usage:

```
number("22"); // returns 22.0  
number("3.1415926"); // returns 3.1415926  
number("0.000001"); // returns 0.000001  
number("infinite"); // returns INFINITE  
number(""); // returns 0.0
```

3.7.26 number (String, Symbol)

number converts a string to a number using the named format. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: the Number representation of the formatted String

Parameters: - a String

- the named format (a Symbol) to use

Example Usage:

```
3.7.27 percentage (Integer)  
percentage converts an integer to a percentage. Percentages are one-based (i.e., 1 = 100%).
```

Returns: the Percentage representation of the Integer

Parameters: - an Integer

Example Usage:

```
percentage(1); // returns 100%  
percentage(-10); // returns -1000%
```

3.7.28 percentage (Number)
percentage converts a number to a percentage. Percentages are one-based (i.e., 1 = 100%).

Returns: the Percentage representation of the Number

Parameters: - a Number

Example Usage:

```
percentage(0.1); // returns 10%  
percentage(-1.01); // returns -101%
```

3.7.29 percentage (Quantity)
percentage converts a unitless quantity to a percentage. Percentages are one-based (i.e., 1 = 100%).

Returns: the Percentage representation of the Quantity or an error if the quantity is not unitless.

Parameters: - a unitless Quantity

Example Usage:

```
percentage(quantity(2)); // returns 200%  
percentage(quantity(0.01)); // returns 1%  
percentage(quantity("1 m")); // returns an error
```

3.7.30 percentage (String)
percentage converts a string to a percentage using the default format.

Returns: the Percentage representation of the String

Parameters: - a String; the string must contain a numeric value

Example Usage:

```
percentage("0"); // returns 0%  
percentage("10.0"); // returns 1000%
```

```
percentage("infinite"); // returns INFINITE  
percentage(""); // returns an error
```

3.7.31 percentage (String, Symbol)
percentage converts a string to a percentage using the named format. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: the Percentage representation of the String

Parameters: - a String; the string must contain a numeric value
- the named format (a Symbol) to use

Example Usage:

3.7.32 power (Number, Number)
power computes number^{power} (number raised to a power).

Returns: a number raised to a power

Parameters: - a Number to be raised
- a power (Number) to raise to

Example Usage:

```
power(10.0, 3.0); // returns 1000.0  
power(0.0, 0.0); // returns 1.0  
power(2.0, -2.5); // returns 0.176777
```

3.7.33 quantity (String)
quantity converts a computed string to a unitless quantity.

Returns: a unitless Quantity representation of the String

Parameters: - a String; the string must contain a numeric value

Example Usage:

```
quantity("3.1415926"); // returns 3.1415926
```

3.7.34 quantity (Integer)
quantity converts an integer to a unitless quantity.

Basic Functions

quantity (Number)

Returns: a unitless Quantity representation of the Integer

Parameters: - an Integer

Example Usage:

```
quantity(1); // returns 1  
quantity(10); // returns 10
```

3.7.35 quantity (Number)

quantity converts a number to a unitless quantity.

Returns: a unitless Quantity representation of the Number

Parameters: - a Number

Example Usage:

```
quantity(20); // returns 20  
quantity(-2.2); // returns -2.2
```

3.7.36 quantity (String)

quantity converts a string to a quantity using the default format. Quantities can have measurement units (e.g., feet, meters, centimeters, inches, kilograms, etc.), but this is not required. The measurement units must be a valid measure.

Returns: the Quantity representation of the String

Parameters: - a String, the string must contain a numeric value followed by an optional measurement unit

Example Usage:

```
quantity("22 m"); // returns 22 m  
quantity("-5 kg"); // returns -5 kg  
quantity("infinite"); // returns INFINITE  
quantity("0 any"); // returns an error (any is not a valid  
// measurement unit)  
quantity(""); // returns an error  
quantity("abc def"); // returns an error
```

Basic Functions

quantity (String, Symbol)

3.7.37 quantity (String, Symbol)

quantity converts a string to a quantity using the named format. Quantities can have measurements (e.g., feet, meters, centimeters, inches, kilograms, etc.), but this is not required. The measurements must be a valid. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: the Quantity representation of the String in the named format

Parameters: - a String, the string must contain a numeric value followed by an optional measurement unit
- the named format (a Symbol) to use

Example Usage:

3.7.38 quantity (Time)

quantity converts a time to a quantity.

Returns: a Quantity representation of the Time

Parameters: - a Time

Example Usage:

```
quantity(time("3600 sec")); // returns 3600 sec  
quantity(time("2 hr")); // returns 2 hr
```

3.7.39 round (Number)

round returns the nearest integer to the given number.

Returns: the nearest Integer

Parameters: - a Number

Example Usage:

```
round(9.9); // returns 10  
round(0); // returns 0  
round(-3.3); // returns -3
```

3.7.40 round (Number, Number)

round returns the nearest multiple of a number to a given number.

Returns: the nearest multiple of a Number

Parameters: - a Number
- a non-negative Number to multiply

Example Usage:

```
round(9.9, 4.2); // returns 8.4  
round(0, 5); // returns 0  
round(5, 0); // returns an error  
round(-4.6, -1.0); // returns an error
```

3.7.41 round_down (Number)

round_down returns the largest integer smaller than a number. Rounds towards negative infinity.

Returns: -oo < the largest Integer <= Number

Parameters: - a Number

Example Usage:

```
round_down(9.9); // returns 9  
round_down(0.1); // returns 0  
round_down(-0.1); // returns -1
```

3.7.42 round_down (Number, Number)

round_down returns the largest multiple of a number not greater than a given number. Rounds towards negative infinity.

Returns: -oo < the largest multiple of a Number <= given Number

Parameters: - a Number
- a non-negative Number to multiply

Example Usage:

```
round_down(9.9, 4.2); // returns 8.4  
round_down(0, 5); // returns 0  
round_down(5, 0); // returns an error  
round_down(-2.7, 1.5); // returns -3.0  
round_down(-4.6, -1.0); // returns an error
```

3.7.43 round_in (Number)

round_in returns the next integer towards zero than a number.

Returns: the next Integer towards zero <= Number

Parameters: - a Number

Example Usage:

```
round_in(9.9); // returns 9  
round_in(0, 1); // returns 0  
round_in(-0, 1); // returns 0
```

3.7.44 round_in (Number, Number)

round_in returns the next multiple of a number towards zero than a given number.

Returns: the next multiple of a Number towards zero >= given Number

Parameters: - a Number
- a non-negative Number to multiply

Example Usage:

```
round_in(9.9, 4.2); // returns 8.4  
round_in(0, 5); // returns 0  
round_in(5, 0); // returns an error  
round_in(-2.7, 1.5); // returns -1.5  
round_in(-4.6, -1.0); // returns an error
```

3.7.45 round_out (Number)

round_out returns the next integer away from zero than a number.

Returns: the next Integer away from zero >= Number

Parameters: - a Number

Example Usage:

```
round_out(9.9); // returns 10  
round_out(0, 1); // returns 1  
round_out(-0, 1); // returns -1
```

3.7.46 round_out (Number, Number)

round_out returns the next multiple of a number away from zero than a given number.

Returns: the next multiple of a Number away from zero >= given Number

Parameters: - a Number
- a non-negative Number to multiply

Example Usage:

```
round_out(9.9, 4.2); // returns 12.6
round_out(0.5); // returns 0
round_out(5.0); // returns an error
round_out(-2.7, 1.5); // returns -3.0
round_out(-4.6, -1.0); // returns an error
```

3.7.47 round_up (Number)

round_up returns the smallest integer larger than a number. Rounds towards positive infinity.

Returns: +∞ > the smallest Integer >= Number

Parameters: - a Number

Example Usage:

```
round_up(9.9); // returns 10
round_up(0.1); // returns 1
round_up(-0.1); // returns 0
```

3.7.48 round_up (Number, Number)

round_up returns the smallest multiple of a number not less than a given number. Rounds towards positive infinity.

Returns: +∞ > the smallest multiple of a Number >= given Number

Parameters: - a Number
- a non-negative Number to multiply

Example Usage:

```
round_up(9.9, 4.2); // returns 12.6
round_up(0.5); // returns 0
round_up(5.0); // returns an error
round_up(-2.7, 1.5); // returns -1.5
round_up(-4.6, -1.0); // returns an error
```

3.7.49 sin (Number)

sin computes the trigonometric sine of an angle, which is taken to be in radians.

Returns: the sine of an angle (a Number)

Parameters: - an angle (a Number) specified in radians

Example Usage:

```
sin(0.0); // returns 0.0
sin(3.141592654); // returns 0.0
sin(-0.5); // returns -0.479426
```

3.7.50 sqrt (Number)

sqrt computes the nonnegative square root of a number.

Returns: the nonnegative square root of a number

Parameters: - a Number; must be >= zero

Example Usage:

```
sqrt(0.0); // returns 0.0
sqrt(4.0); // returns 2.0
sqrt(-2.0); // returns an error
```

3.7.51 tan (Number)

tan computes the trigonometric tangent of an angle, which is taken to be in radians.

Returns: the tangent of an angle (a Number)

Parameters: - an angle (a Number) specified in radians

Example Usage:

```
tan(0.0); // returns 0.0
tan(1.570796327); // returns -INFINITE
tan(-0.5); // returns -0.546302
```

3.7.52 time (Quantity)

time converts a quantity to a time. The quantity must be expressed in valid time measurement units (sec, min, or hr).

Returns: the Time representation of the Quantity

Parameters: - a Quantity; the quantity must be expressed in a valid time measure

Example Usage:

```
time(quantity("3 sec")); // returns 00:00:03
time(quantity("-5 min")); // returns -00:05:00
time(quantity("10")); // returns an error
time(quantity("1 year")); // returns an error
```

3.7.53 time (String)

time converts a string to a time using the default format. Times must specify measurement units. The valid units are sec, min, and hr.

Returns: the Time representation of the String

Parameters: - a String; the string must contain a numeric value followed by a valid time measure

Example Usage:

```
time("22 sec"); // returns 00:00:23
time("-5 min"); // returns -00:05:00
time("3 hr"); // returns 03:00:00
time("infinite"); // returns INFINITE
time("22"); // returns an error
time("0 any"); // returns an error (any is not a valid unit)
time(""); // returns an error
time("abc def"); // returns an error
```

3.7.54 time (String, Symbol)

time converts a string to a time using the named format. Times must specify measurement units. The valid units are sec, min, and hr. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: the Time representation of the String in the named format

Parameters: - a String; the string must contain a numeric value followed by a valid time measure
- the named format (a Symbol) to use

Example Usage:

3.7.55 units (Quantity)

units returns the quantity that represents one unit of a quantity.

Returns: quantity that represents one measure unit of a Quantity

Parameters: - a Quantity

Example Usage:

```
units("0"); // returns "1"
units("-2.5 kg"); // returns "1 kg"
units("25 par/hr"); // returns "1 par/hr"
```

3.8 Random Functions

Random Number generation functions.

3.8.1 rand_integer (Integer, Integer)

rand_integer generates an evenly distributed random integer between two given integers, both inclusive. Use rand_seed for initializing the pseudo-random-number generator used by this function.

Returns: a random Integer between two given values (both inclusive)

Parameters: - an inclusive Integer
- an inclusive Integer

Example Usage:

```
rand_integer(0, 10); // returns a random integer >= 0 and <= 10
rand_integer(-1, 0); // returns either -1 or 0
rand_integer(1, 1); // returns 1
rand_integer(1, 0); // returns either 1 or 0
```

3.8.2 rand ()

rand generates an evenly distributed random number between 0 and 1, both inclusive. Use rand_seed for initializing the pseudo-random-number generator used by this function.

Returns: a random Number between 0 and 1 (both inclusive).

Parameters: - N/A

Example Usage:

Basic Functions	rand (Number, Number)
-----------------	-----------------------

rand(): // returns a number between 0 and 1

3.8.3 rand (Number, Number)

rand generates an evenly distributed random number between two given numbers, both inclusive. Use rand_seed for initializing the pseudo-random-number generator used by this function.

Returns: a random Number between two given values (both inclusive)

Parameters: - an inclusive Number
- an inclusive Number

Example Usage:

```
rand(0.5, 5.3); // returns a random integer >= 0.5 and <= 5.3
rand(1.0, 1.0); // returns 1.0
rand(2.0, -2.0); // returns a random number >= -2.0 and <= 2.0
```

3.8.4 rand_seed (Number)

rand_seed initializes the pseudo-random-number generator using the given number as a seed. This allows for reproducibility of a series of random numbers generated from the rand functions.

Returns: N/A

Parameters: - a Number to be used as a seed

Example Usage:

```
rand_seed(39.0);
```

3.9 Quantity_Range Functions

A Quantity_Range is defined by a minimum quantity and a maximum quantity, both inclusive. The functions below can be used to create, query, and manipulate Quantity_Ranges.

3.9.1 intersects (Quantity_Range, Quantity_Range)

intersects determines if two quantity ranges overlap. The quantity ranges must have the same measurement units.

Returns: True if the two quantity ranges overlap, False otherwise.

Parameters: - a Quantity_Range

Basic Functions	within (Quantity, Quantity_Range)
-----------------	------------------------------------

- a Quantity_Range

Example Usage:

Given...

```
a = quantity_range("1 ft / 3 ft");
b = quantity_range("2 ft / 4 ft");
c = quantity_range("4 ft / 6 ft");
d = quantity_range("1 hr / 3 hr");
```

Then...

```
intersect(a, b); // returns True
intersect(a, c); // returns False
intersect(b, c); // returns True
intersect(a, d); // returns an error
```

3.9.2 within (Quantity, Quantity_Range)

within determines if a quantity is within a quantity range. The quantity and quantity range must have the same measurement units.

Returns: True if the quantity is within the quantity range, False otherwise.

Parameters: - a Quantity
- a Quantity_Range

Example Usage:

Given...

```
a = quantity_range("1 ft / 3 ft");
b = quantity_range("2 ft / 4 ft");
c = quantity_range("4 ft / 6 ft");
d = quantity_range("1 hr / 3 hr");
```

Then...

```
within("2 ft", b); // returns True
within("2 ft", c); // returns True
within("2 ft", d); // returns False
within("2 ft", d); // returns an error
```

3.9.3 within (Quantity_Range, Quantity_Range)

within determines if a quantity range is within another quantity range. The quantity ranges must have the same measurement units.

Returns: True if the first quantity range is within the second quantity range. False otherwise.

Parameters: - a Quantity_Range
- a Quantity_Range

Example Usage:

Given...
a = quantity_range("1 ft / 3 ft");
b = quantity_range("5 ft / 6 ft");
c = quantity_range("4 ft / 8 ft");
d = quantity_range("1 hr / 3 hr");

Then...
within(a, b); // returns False
within(b, c); // returns True
within(c, b); // returns False
within(a, d); // returns an error

3.9.4 interval (Quantity_Range)

interval returns or sets the interval of a quantity range. When querying, it returns the interval as a quantity. When setting, it changes the maximum quantity of the quantity range.

Returns: a Quantity with the same measurement units as the Quantity_Range

Parameters: - a Quantity_Range

Example Usage:

Given...
a = quantity_range("1 hr / 3 hr");
b = quantity_range(0, "infinite");
c = quantity_range("30 sec / 180 sec");

Then...

3.9.5 limit (Quantity, Quantity_Range)

limit either returns a quantity if it is within a range or the closest boundary of the range. The quantity and the range must have the same measurement units.

Returns: a Quantity in the quantity range

Parameters: - a Quantity
- a Quantity_Range

Example Usage:

Given...
a = quantity_range("1 hr / 3 hr");

Then...
limit("4 hr", a); // returns 4 hr
limit("2 hr", a); // returns 2 hr
limit("2 kg", a); // returns an error

3.9.6 limit (Quantity_Range, Quantity_Range)

limit returns a quantity range which represents the intersection of two quantity ranges. If the two ranges do not intersect, then the maximum value of the first range is returned as a range. The two quantity ranges must have the same measurement units.

Returns: the intersection Quantity_Range

Parameters: - a Quantity_Range
- a Quantity_Range

Example Usage:

Given...
a = quantity_range("1 ft / 3 ft");
b = quantity_range("2 ft / 4 ft");
c = quantity_range("4 ft / 6 ft");
d = quantity_range("1 hr / 3 hr");

Then...

```
limit(a, b); // returns 2 ft / 3 ft
limit(a, c); // returns 3 ft / 3 ft
limit(b, c); // returns 4 ft / 4 ft
limit(a, d); // returns an error
```

3.9.7

max (Quantity_Range)

max sets or returns the maximum quantity of a quantity range. If setting and the new maximum is less than the current minimum, then the range's minimum is changed to match the new maximum.

Returns: the maximum Quantity of a Quantity_Range

Parameters: - a Quantity_Range

Example Usage:

Given...

```
a = quantity_range("1 hr / 3 hr");
b = quantity_range(0, "infinite");
c = quantity_range("30 sec / 180 sec");
```

Then...

```
max(a); // returns 3 hr
max(b); // returns INFINITE
c.max = "20 sec"; // changes the range to 20 sec / 20 sec
```

3.9.8

min (Quantity_Range)

min sets or returns the minimum quantity of a quantity range. If setting and the new minimum is greater than the current maximum, then the range's maximum is changed to match the new minimum.

Returns: the minimum Quantity of a Quantity_Range

Parameters: - a Quantity_Range

Example Usage:

Given...

```
a = quantity_range("1 hr / 3 hr");
```

```
b = quantity_range(0, "infinite");
c = quantity_range("30 sec / 180 sec");
```

Then...

```
min(a); // returns 1 hr
min(b); // returns 0
c.min = "200 sec"; // changes the range to 200 sec / 200 sec
```

3.9.9

enclose (Quantity_Quantity_Range)

enclose defines a new quantity range which includes a specified quantity. If the quantity is already within the given range, then the new range will match the given range. If quantity's measurement unit does not match the range's measurement unit, the original range is returned.

Returns: a Quantity_Range with Quantity within it or the original Quantity_Range if the Quantity has different measurement units

Parameters: - a Quantity to include
- the Quantity_Range to stretch

Example Usage:

Given...

```
a = quantity_range("1 hr / 3 hr");
b = quantity_range(0, "infinite");
c = quantity_range("30 sec / 180 sec");
```

Then...

```
enclose("5 hr", a); // returns 1 hr / 5 hr
enclose(100, b); // returns 0 / INFINITE
enclose("10 sec", c); // returns 10 sec / 180 sec
enclose("5 kg", a); // returns 1 hr / 3 hr
```

3.9.10 enclose (Quantity_Range, Quantity_Range)

enclose defines a new quantity range which includes the first range within the second range. If the first range is already within the second range, then the new range will match the second range. If the range's measurement units do not match, the second range is returned.

Returns: a Quantity_Range with the first Quantity_Range enclosed with the second Quantity_range or the second Quantity_Range if the first Quantity has different measurement units

Parameters: - a Quantity_Range to include
- the Quantity_Range to stretch

Example Usage:

Given...
a = quantity_range("1 hr / 3 hr");
b = quantity_range("4 hr / 9 hr");
c = quantity_range("5 hr / 7 hr");
d = quantity_range("1 sec / 10 sec");

Then...

enclose(a, b); // returns 1 hr / 9 hr
enclose(b, c); // returns 4 hr / 9 hr
enclose(d, a); // returns 1 hr / 3 hr

3.9.11 quantity_interval (Quantity, Quantity)

quantity_interval creates a quantity_range using the first quantity as an anchor point and the second quantity as the interval from the anchor. If the second quantity is positive, the first quantity is the new range's minimum; if it is negative, the first quantity is the new range's maximum. The quantities must have the same measurement units.

Returns: a new Quantity_Range

Parameters: - the anchor Quantity
- the interval Quantity

Example Usage:

quantity_interval("1 kg", "3 kg"); // returns 1 kg / 4 kg
quantity_interval("1 kg", "0 kg"); // returns 1 kg / 1 kg
quantity_interval("4 kg", "-2 kg"); // returns 2 kg / 4 kg
quantity_interval("1 kg", "2 hr"); // returns an error

3.9.12 quantity_range (Quantity, Quantity)

quantity_range creates a new quantity_range given two quantities. The first quantity must be equal to or small than the second quantity. The quantities must have the same measurement unit.

Returns: a new Quantity_Range

Parameters: - a Quantity; must be <= second Quantity
- a Quantity

Example Usage:

quantity_range("1 hr", "4 hr"); // returns 1 hr / 4 hr
quantity_range("5 kg", "2 kg"); // returns an error
quantity_range("1 hr", "2 kg"); // returns an error

3.9.13 quantity_range (String)

quantity_range converts a string into a quantity_range using the default format. The string must contain two quantities which have the same measurement units and are separated by the string "<". The smaller quantity must be first.

Returns: the Quantity_Range representation of the String

Parameters: - a String; the string must contain two quantities of the same measurement unit with the smaller one preceding the larger.

Example Usage:

quantity_range("1 hr / 4 hr"); // returns 1 hr / 4 hr
quantity_range("1 hr 4 hr"); // returns 1 hr / 1 hr
quantity_range("5 kg / 2 kg"); // returns an error
quantity_range("1 hr / 2 kg"); // returns an error

3.9.14 quantity_range (String, Symbol)

quantity_range converts a string into a quantity_range using the named format. The string must contain two quantities which have the same measurement units. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: the Quantity_Range representation of the String

Parameters: - a String; the string must contain two quantities of the same measurement unit.
- the named format (a Symbol) to use

Example Usage:

3.9.15 set_max (Quantity_Range, Quantity)

set_max sets the maximum quantity of a quantity range. If the new maximum is less than the range's minimum, then the minimum is also set to the new maximum value. The quantity must have the same measurement units as the range.

Returns: the new maximum Quantity of Quantity_Range

Parameters: - a Quantity_Range
- the maximum Quantity for the range

Example Usage:

Given...

```
a = quantity_range("1 hr / 3 hr");
b = quantity_range(0, "infinite");
```

Then...

```
set_max(a, "2 hr"); // returns 1 hr / 2 hr
set_max(b, -2); // returns -2 / -2
set_max("5 kg", a); // returns an error
```

3.9.16 set_min (Quantity_Range, Quantity)

set_min sets the minimum quantity of a quantity range. If the new minimum is greater than the range's maximum, then the maximum is also set to the new minimum value. The quantity must have the same measurement units as the range.

Returns: the new minimum Quantity of Quantity_Range

Parameters: - a Quantity_Range
- the minimum Quantity for the range

Example Usage:

Given...

```
a = quantity_range("1 hr / 3 hr");
b = quantity_range(0, "infinite");
```

Then...

```
set_min(a, "5 hr"); // returns 5 hr / 5 hr
set_min(b, 100); // returns 100 / INFINITE
```

```
set_min("2 kg", a); // returns an error
```

3.9.17 set_interval (Quantity_Range, Quantity)

set_interval adjusts the interval of a quantity range, increasing or decreasing the maximum quantity as necessary. The given quantity must be greater than zero and must have the same measurement units as the range.

Returns: an adjusted Quantity_Range

Parameters: - a Quantity_Range
- a Quantity which defines a new range interval; must be > 0

Example Usage:

Given...

```
a = quantity_range("1 hr / 3 hr");
b = quantity_range(0, "infinite");
```

Then...

```
set_interval(a, "5 hr"); // returns 1 hr / 6 hr
set_interval(b, 1); // returns 0 / 1
set_interval(a, "-1 hr"); // returns an error
set_interval(a, "3 kg"); // returns an error
```

3.10 Numeric List Functions

Numeric lists are sequential collections of numbers, integers, quantities, percentages, or times. The statistical functions documented within this section apply exclusively to numeric lists. They all return a single number, integer, quantity, percentage, or time based on the kind of numeric list they are given.

You can find the minimum or maximum in the List, or compute the sum, average, or standard deviation of the elements of the numeric list.

3.10.1 average (List(Integer))

average computes the average (sum / # elements) of all elements in an integer list. The list must have at least one element.

Returns: the average (a Number) of all the integers in a list

Parameters: - an integer List

Example Usage:

```
average(list(1, 2, 3)); // returns 2.5  
average(list()); // returns an error  
list(-3, 8, -5).average; // returns 0
```

3.10.2 average (List(Number))

average computes the average (sum / # elements) of all elements in a number list. The list must have at least one element.

Returns: the average (a Number) of all the numbers in a list

Parameters: - a number List

Example Usage:
average(list(0.4, 1.6, 2.9)); // returns 1.63
average(list()); // returns an error
list(-3.3, 8.1, -5.7).average; // returns -0.3

3.10.3 average (List(Percentage))

average computes the average (sum / # elements) of all elements in a percentage list. The list must have at least one element.

Returns: the average of all the percentages in a list

Parameters: - a percentage List

Example Usage:
average(list(25%, 67%, 125%)); // returns 72.33%
average(list()); // returns an error
list(-50%, 10%, 65%).average; // returns 8.33%

3.10.4 average (List(Quantity))

average computes the average (sum / # elements) of all elements in a quantity list. The list must have at least one element.

Returns: the average of all the quantities in a list

Parameters: - a quantity List

Example Usage:
average(list(25 kg, 67 kg, 125 kg)); // returns 72.33 kg
average(list()); // returns an error

```
list(-50 kg, 10 kg, 65 kg).average; // returns 8.33 kg
```

3.10.5 average (List(Time))

average computes the average (sum / # elements) of all elements in a time list. The list must have at least one element.

Returns: the average of all the times in a list

Parameters: - a time List

Example Usage:
average(list(25 sec, 67 sec, 125 sec)); // returns 72.33 sec
average(list()); // returns an error
list(-50 sec, 10 sec, 65 sec).average; // returns 8.33 sec

3.10.6 max (List(Integer))

max returns the largest element of an integer list. The list must have at least one element.

Returns: the largest Integer in a list

Parameters: - an integer List

Example Usage:
max(list(1, 2, 3)); // returns 3
max(list()); // returns an error
list(-3, 8, -5).max; // returns 8

3.10.7 max (List(Number))

max returns the largest element of a number list. The list must have at least one element.

Returns: the largest Number in a list

Parameters: - a number List

Example Usage:
max(list(0.4, 1.6, 2.9)); // returns 2.9
max(list()); // returns an error
list(-3.3, 8.1, -5.7).max; // returns 8.1

3.10.8 max (List(Percentage))

max returns the largest element of a percentage list. The list must have at least one element.

Returns: the largest Percentage in a list

Parameters: - a percentage List

Example Usage:

```
max(list(25%, 67%, 125%)); // returns 125%
max(list()); // returns an error
list(-50%, 10%, 65%).max; // returns 65%
```

3.10.9 max (List(Quantity))

max returns the largest element of a quantity list. The list must have at least one element.

Returns: the largest Quantity in a list

Parameters: - a quantity List

Example Usage:

```
max(list(25 kg, 67 kg, 125 kg)); // returns 125 kg
max(list()); // returns an error
list(-50 kg, 10 kg, 65 kg).max; // returns 65 kg
```

3.10.10 max (List(Time))

max returns the largest element of a time list. The list must have at least one element.

Returns: the largest Time in a list

Parameters: - a time List

Example Usage:

```
max(list(25 sec, 67 sec, 125 sec)); // returns 125 sec
max(list()); // returns an error
list(-50 sec, 10 sec, 65 sec).max; // returns 65 sec
```

3.10.11 min (List(Integer))

min returns the smallest element of an integer list. The list must have at least one element.

Returns: the smallest Integer in a list

Parameters: - an integer List

Example Usage:

```
min(list(1, 2, 3)); // returns 1
min(list()); // returns an error
list(3, 8, -5).min; // returns -5
```

3.10.12 min (List(Number))

min returns the smallest element of a number list. The list must have at least one element.

Returns: the smallest Number in a list

Parameters: - a number List

Example Usage:

```
min(list(0.4, 1.6, 2.9)); // returns 0.4
min(list()); // returns an error
list(3.3, 8.1, -5.7).min; // returns -5.7
```

3.10.13 min (List(Percentage))

min returns the smallest element of a percentage list. The list must have at least one element.

Returns: the smallest Percentage in a list

Parameters: - a percentage List

Example Usage:

```
min(list(25%, 67%, 125%)); // returns 25%
min(list()); // returns an error
list(-50%, 10%, 65%).min; // returns -50%
```

3.10.14 min (List(Quantity))

min returns the smallest element of a quantity list. The list must have at least one element.

Returns: the smallest Quantity in a list

Basic Functions	min (List[Time])
-----------------	------------------

Parameters: - a quantity List

Example Usage:

```
min(list(25 kg, 67 kg, 125 kg)); // returns 25 kg
min(list(0); // returns an error
list(-50 kg, 10 kg, 65 kg).min; // returns -50 kg
```

3.10.15 min (List[Time])

min returns the smallest element of a time list. The list must have at least one element.

Returns: the smallest Time in a list

Parameters: - a time List

Example Usage:

```
min(list(25 sec, 67 sec, 125 sec)); // returns 25 sec
min(list(0); // returns an error
list(-50 sec, 10 sec, 65 sec).min; // returns -50 sec
```

3.10.16 product (List[Integer])

product multiplies together all the elements in an integer List. The list must have at least one element.

Returns: the product of all the integers in a list

Parameters: - an Integer List

Example Usage:

```
product(list(1, 2, 3)); // returns 6.0
product(list(0); // returns an error
list(-3, 8, -5).product; // returns 120.0
```

3.10.17 product (List[Number])

product multiplies together all the elements in a number List. The list must have at least one element.

Returns: the product of all the numbers in a list

Parameters: - a number List

Basic Functions	product (List[Percentage])
-----------------	----------------------------

Example Usage:

```
product(list(0.4, 1.6, 2.9)); // returns 1.856
product(list(0); // returns an error
list(-3.3, 8.1, -5.7).product; // returns 152.361
```

3.10.18 product (List[Percentage])

product multiplies together all the elements in a percentage List. The list must have at least one element.

Returns: the product of all the percentages in a list

Parameters: - a percentage List

Example Usage:

```
product(list(25%, 67%, 125%)); // returns 20.9375%
product(list(0); // returns an error
list(-50%, 10%, 65%).product; // returns -3.25%
```

3.10.19 product (List[Quantity])

product multiplies together all the elements in a quantity List. The list must have at least one element.

Returns: the product of all the quantities in a list

Parameters: - a quantity List

Example Usage:

```
product(list(25 kg, 67 kg, 125 kg)); // returns 209375 kg
product(list(0); // returns an error
list(-50 kg, 10 kg, 65 kg).product; // returns -32500 kg
```

3.10.20 stdev (List[Integer])

stdev computes the standard deviation of all elements in an integer list. The list must contain at least one element.

Returns: the standard deviation of all the integers in a list

Parameters: - an integer List

Example Usage:

```
stdev(list(1, 2, 3)); // returns 1.0
```

Basic Functions**stdev (List(Number))**

stdev(0); // returns an error
 list(-3, 8, -5).stdev; // returns 7.0

3.10.21 stdev (List(Number))

stdev computes the standard deviation of all elements in a number list. The list must contain at least one element.

Returns: the standard deviation of all the numbers in a list

Parameters: - a number List

Example Usage:

stdev(list(0.4, 1.6, 2.9)); // returns 1.25033
 stdev(list(0)); // returns an error
 list(-3.3, 8.1, -5.7).stdev; // returns 7.37292

3.10.22 stdev (List(Percentage))

stdev computes the standard deviation of all elements in a percentage list. The list must contain at least one element.

Returns: the standard deviation of all the percentages in a list

Parameters: - a percentage List

Example Usage:

stdev(list(25%, 67%, 125%)); // returns 0.502129
 stdev(list(0)); // returns an error
 list(-50%, 10%, 65%).stdev; // returns 0.575181

3.10.23 stdev (List(Quantity))

stdev computes the standard deviation of all elements in a quantity list. The list must contain at least one element.

Returns: the standard deviation of all the quantities in a list

Parameters: - a quantity List

Example Usage:

stdev(list(25 kg, 67 kg, 125 kg)); // returns 50.21 kg
 stdev(list(0)); // returns an error
 list(-50 kg, 10 kg, 65 kg).stdev; // returns 57.52 kg

Basic Functions**stdev (List(Time))**

3.10.24 stdev (List(Time))
 stdev computes the standard deviation of all elements in a time list. The list must contain at least one element.

Returns: the standard deviation of all the times in a list

Parameters: - a time List

Example Usage:

stdev(list(25 sec, 67 sec, 125 sec)); // returns 50.21 sec
 stdev(list(0)); // returns an error
 list(-50 sec, 10 sec, 65 sec).stdev; // returns 57.52 sec

3.10.25 sum (List(Integer))

sum adds together all the elements in an integer list. The list must have at least one element.

Returns: the sum of all the integers in a list

Parameters: - an integer List

Example Usage:

sum(list(1, 2, 3)); // returns 6
 sum(list(0)); // returns an error
 list(-3, 8, -5).sum; // returns 0

3.10.26 sum (List(Number))

sum adds together all the elements in a number list. The list must have at least one element.

Returns: the sum of all the numbers in a list

Parameters: - a number List

Example Usage:

sum(list(0.4, 1.6, 2.9)); // returns 4.9
 sum(list(0)); // returns an error
 list(-3.3, 8.1, -5.7).sum; // returns -0.9

3.10.27 sum (List(Percentage))

sum adds together all the elements in a percentage List. The list must have at least one element.

Returns: the sum of all the percentages in a list

Parameters: - a percentage List

Example Usage:

```
sum(list(25%, 67%, 125%)); // returns 217%
sum(list()); // returns an error
list(-50%, 10%, 65%),sum; // returns 25%
```

3.10.28 sum (List(Quantity))

sum adds together all the elements in a quantity List. The list must have at least one element.

Returns: the sum of all the quantities in a list

Parameters: - a quantity List

Example Usage:

```
sum(list(25 kg, 67 kg, 125 kg)); // returns 217 kg
sum(list()); // returns an error
list(-50 kg, 10 kg, 65 kg),sum; // returns 25 kg
```

3.10.29 sum (List(Time))

sum adds together all the elements in a time List. The list must have at least one element.

Returns: the sum of all the times in a list

Parameters: - a time List

Example Usage:

```
sum(list(25 sec, 67 sec, 125 sec)); // returns 217 sec
sum(list()); // returns an error
list(-50 sec, 10 sec, 65 sec),sum; // returns 25 sec
```

3.11 Date Functions

A date is a particular point in time, past, present, or future, with precision to one second. For example,

1996 Mar 15 14:10:02 - March 15, 1996 at 2:10PM

Dates must be in the range from 1970 to 2050. Dates do not require a time, but including a time with a date further refines the exact moment the date represents. Dates can be subtracted, yielding the time duration between the two dates. Two special dates are supported:

infinite_past ("----") - beginning of time in finite_future ("+++++") - end of time

Dates can be created and queried with the following functions.

3.11.1 max (List(Date))

max returns the largest element of a date list. The list must have at least one element.

Returns: the largest Date in a list

Parameters: - a date List

Example Usage:

```
max(list(date("95/01/10 00:00"), date("95/01/05 00:00"))); // returns "95/01/10 00:00"
```

3.11.2 min (List(Date))

min returns the smallest element of a date list. The list must have at least one element.

Returns: the smallest Date in a list

Parameters: - a date List

Example Usage:

```
min(list(date("95/01/10 00:00"), date("95/01/05 00:00"))); // returns "95/01/05 00:00"
```

3.11.3 date (String)

date converts a string to a date using the default format. The default format is "YY-MM-DD hh:mm:ss".

Returns: the Date representation of the String

Basic Functions	date (String, Symbol)
-----------------	-----------------------

Parameters: - a String, the string must contain a valid date and time, specified in seconds

Example Usage:

```
date("96-1-1 00:00:00"); // returns 96-01-01 00:00:00
date("97-2-29 12:00:00"); // returns an error
date("69-12-31 23:59:59"); // returns an error
date("51-1-1 00:00:00"); // returns an error
date("96-7-4"); // returns an error
date(""); // returns an error
date("abcde"); // returns an error
```

3.11.4 date (String, Symbol)

date converts a string to a date using the named format. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: the Date representation of the String

Parameters: - a String, the string must be able to be parsed using the named format
- the named format (a Symbol) to use

Example Usage:

```
Given a 'minutes' format, specified as "MM-DD-YY hr:mm", then
@- date("1-1-96 00:00", minutes); // returns 96-01-01 00:00:00
date("9-9-99", minutes); // returns an error
```

Given a 'no_time' format, specified as "MM-DD-YY", then

```
@- date("7-4-96", no_time); // returns 96-07-04 00:00:00
date("1-1-96 18:30:00", no_time); // returns an error
```

3.11.5 now()

now returns the current date and time.

Returns: The current date and time

Parameters: - N/A

Example Usage:

Basic Functions	start_of_day (Date)
-----------------	---------------------

now() // returns the current date and time

3.11.6 start_of_day (Date)

start_of_day returns the start of the day of a date in the local time zone.

Returns: The start of the day for the given date

Parameters: - a Date

Example Usage:

```
start_of_day("00-1-1 12:00:00"); // returns 00-01-01 00:00:00
```

3.11.7 start_of_day (Date, Integer)

start_of_day returns the start of the day in the local time zone of a future date specified as a certain number of days beyond a given date.

Returns: The start of the day of a future date X number of days in the future

Parameters: - a Date
- the number of days (Integer) in the future.
Negative numbers indicate days in the past.

Example Usage:

```
start_of_day("00-1-28 14:30:23", 5); // returns 00-02-02 00:00:00
```

```
date_range(day_no, start_of_day(day_no, 1)) // a 1-day long date_range
Note: date_range(day_no, day_no + "24 hr") produces different results when day-
light-savings time changes.
```

3.11.8 start_of_month (Date)

start_of_month returns the start of the month of a date in the local time zone.

Returns: The start of the month for the given date

Parameters: - a Date

Example Usage:

```
start_of_month("00-1-16 12:00:00"); // returns 00-01-01 00:00:00
```


Basic Functions	start_of_month (Date, Integer)
-----------------	--------------------------------

3.11.9 start_of_month (Date, Integer)

start_of_month returns the start of the month in the local time zone of a future or past date specified as a certain number of months beyond or prior to a given date.

Returns: The start of the month of a future or past date X number of months in the future or past

Parameters: - a Date
- the number of months (Integer) in the future (if positive) or past (if negative)

Example Usage:

```
start_of_month("00-01-28 14:30:23", 5); // returns 00-06-01 00:00:00
start_of_month("97-02-01 00:00:00", -1); // returns 97-01-01 00:00:00
```

3.11.10 start_of_week (Date)

start_of_week returns the start of the week of a date in the local time zone. The start of a week is Monday.

Returns: The start of the week for the given date

Parameters: - a Date

Example Usage:

```
start_of_week("00-1-14 12:00:00"); // returns 00-01-10 00:00:00
```

3.11.11 start_of_week (Date, Integer)

start_of_week returns the start of the week in the local time zone of a future or past date specified as a certain number of weeks beyond or prior to a given date.

Returns: The start of the week of a future or past date X number of weeks in the future or past

Parameters: - a Date
- the number of weeks (Integer) in the future (if positive) or past (if negative)

Example Usage:

```
start_of_week("00-1-28 14:30:23", 5); // returns 00-02-28 00:00:00
```

Basic Functions	start_of_year (Date)
-----------------	----------------------

```
start_of_week("00-1-28 14:30:23", -2); // returns 00-01-10 00:00:00
```

3.11.12 start_of_year (Date)

start_of_year returns the start of the year of a date in the local time zone.

Returns: The start of the year for the given date

Parameters: - a Date

Example Usage:

```
start_of_year("00-1-14 12:00:00"); // returns 00-01-01 00:00:00
```

3.11.13 start_of_year (Date, Integer)

start_of_year returns the start of the year in the local time zone of a future or past date specified as a certain number of years beyond or prior to a given date.

Returns: The start of the year of a future or past date X number of years in the future or past

Parameters: - a Date
- the number of years (Integer) in the future (if positive) or past (if negative)

Example Usage:

```
start_of_year("00-1-28 14:30:23", 5); // returns 05-01-01 00:00:00
start_of_year("00-1-28 14:30:23", -2); // returns 98-01-01 00:00:00
```

3.11.14 horizon_date (String)

horizon_date converts a string to Horizon_Date using the default format.

Returns: the Horizon_Date representation of the String

Parameters: - a String

Example Usage:
(Angle brackets indicate optional)
(! symbol indicates either one or the other)
(Time is optional for all the formats and can be specified in a similar way for all: hours:minutes, FORMAT)
(FORMAT in the future at hours:minutes)

Basic Functions	horizon_date (String)
<p>(FORMAT at hour:minutes) where FORMAT can be specified as one of the following:</p> <p>Days - N Days in the future Num_of_days <days day> horizon_date("fifth day"); // returns 5th day</p> <p>Absolute Day of nth Month Num_of_days <days day> <of> Num_of_month monthmonths horizon_date("2nd day of third month"); // returns 2nd of 3rd month</p> <p>Relative day of nth month Num_of_days <daysday> and Num_of_month monthmonths horizon_date("1 day and 4 months"); // returns 1st day and 4th month</p> <p>Absolute day from end of Nth month Num_of_days <daysday> <from> endlast <of> Num_of_month monthmonths horizon_date("2 days from the end of the third month");</p> <p>Nth day of week in future <The> Num_of_occurrence <occurrence of the day> <Day_of_Week> horizon_date("first Tuesday"); // returns 1st Tue</p> <p>Day of week in Nth week of future Day_Of_Week <of> Num_Of_Week WeekWeeksWeeksWeeks <start>start Day_of_Week_start> horizon_date("Tuesday of second week"); // returns Tue of 2nd week, start Mon</p> <p>Nth Day_of_week of Nth month Num_of_Week_Day Day_Of_Week <of> Num_of_Month monthmonths horizon_date("first Thursday of sixth month");</p> <p>Day of week of nth week of nth month Day_Of_Week <of> Num_Of_Week WeekWeeksWeeksWeeks <of> Num_Of_Month monthmonths <start>start Day_Of_Week> horizon_date("Mo of 2nd week, third month"); // returns Mon, 2nd wk, 1st month, start Wed horizon_date("Mo of first week, 1st month, week starts on Tuesday");</p> <p>Use of time in horizon_date:</p>	

Basic Functions	finite (Date)
<p>horizon_date("10:30pm, 2nd day"); horizon_date("2nd day at 10:30pm"); horizon_date("2nd day in the future at 10:30pm");</p> <p>3.11.15 finite (Date) finite returns True if and only if the given date is finite</p> <p>Returns: True when the Date is finite.</p> <p>Parameters: - a Date</p> <p>Example Usage: finite("00-6-14"); // returns True finite("++++"); // returns False</p> <p>3.11.16 day_of_month (Date) day_of_month returns the day of the month in the local time zone. The date must by finite.</p> <p>Returns: the day of the month (1-31)</p> <p>Parameters: - a Date</p> <p>Example Usage: day_of_month("00-1-14 12:00:00"); // returns 14</p> <p>3.11.17 day_of_week (Date) day_of_week returns the day of the week in the local time zone. The date must by finite.</p> <p>Returns: the day of the week (1-7); 1 = Monday</p> <p>Parameters: - a Date</p> <p>Example Usage: day_of_week("00-1-14 12:00:00"); // returns 5</p> <p>3.11.18 day_of_year (Date) day_of_year returns the day of the year in the local time zone. The date must by finite.</p>	

Basic Functions**hour (Date)**

Returns: the day of the year (1-366)

Parameters: - a Date

Example Usage:

```
day_of_year("00-2-14 12:00:00"); // returns 45
```

3.11.19 hour (Date)

hour returns the hour of the day in the local time zone. The date must be finite.

Returns: the hour of the day (0-23)

Parameters: - a Date

Example Usage:

```
hour("00-2-14 12:00:00"); // returns 12
```

3.11.20 minute (Date)

minute returns the minute of the day in the local time zone. The date must be finite.

Returns: the minute of the day (0-59)

Parameters: - a Date

Example Usage:

```
minute("00-2-14 12:34:00"); // returns 34
```

3.11.21 month (Date)

month returns the month of the date in the local time zone. The date must be finite.

Returns: the month of the date (1-12)

Parameters: - a Date

Example Usage:

```
month("00-2-14 12:34:00"); // returns 2
```

Basic Functions**number_of_weeks (Date, Date)****3.11.22 number_of_weeks (Date, Date)**

number_of_weeks returns the number of weeks between two dates in the local time zone. If the first date occurs prior to the second date, the result is negative weeks.

Returns: the number of weeks

Parameters: - the reference Date
- the actual Date

Example Usage:

```
number_of_weeks("00-1-1", "01/1/1"); // returns -52  
number_of_weeks("00-1-1", "99/1/4"); // returns 25
```

3.11.23 second (Date)

second returns the second of the day in the local time zone. The date must be finite.

Returns: the second of the day (0-59)

Parameters: - a Date

Example Usage:

```
minute("00-2-14 12:34:13"); // returns 13
```

3.11.24 week_of_year (Date)

week_of_year returns the number of weeks from the start of the date's year in the local time zone.

Returns: the number of weeks

Parameters: - a Date

Example Usage:

```
week_of_year("00-7-4"); // returns 27
```

3.11.25 year (Date)

year returns the year of the date in the local time zone. The date must be finite.

Returns: the year of the date (1970-2050)

Basic Functions	within_daylight_savings_time (Date)
-----------------	-------------------------------------

Parameters: - a Date

Example Usage:

month("00-2-14 12:34:00"); // returns 2000

3.11.26 within_daylight_savings_time (Date)

within_daylight_saving_time returns TRUE if and only if the given date falls within daylight saving time in the local time zone.

Returns: True when the Date is within daylight savings time; False otherwise.

Parameters: - a Date

Example Usage:

If you are within the US Central time zone, then...

within_daylight_savings_time("00-12-15"); // returns False
within_daylight_savings_time("00-6-14"); // returns True

3.11.27 within_leap_year (Date)

within_leap_year returns True if and only if the given data occurs during a leap year in the local time zone.

Returns: True when the Date is within a leap year; False otherwise

Parameters: - a Date

Example Usage:

within_leap_year("00-6-14"); // returns True
within_leap_year("98-6-14"); // returns False

3.11.28 enclosing_day (Date)

enclosing_day returns a day long time period, beginning at 00:00:00 on the given Date.

Returns: a day long time period (Date_Range)

Parameters: - a Date

Basic Functions	date (Horizon_Date, Date)
-----------------	---------------------------

Example Usage:

enclosing_day("00-6-14 00:00"); // returns 00-6-14 00:00 / 00-6-15 00:00
enclosing_day("00-6-15 18:10"); // returns 00-6-15 00:00 / 00-6-16 00:00

3.11.29 date (Horizon_Date, Date)

Given a Horizon_Date, and a start date, return the Date the given Date.

Returns: a day long time period (Date_Range)

Parameters: - a Date

Example Usage: horizon_date("Tuesday of second week").date("95/02/01 02:00")->95-02-14 00:00

3.11.30 restriction (String)

restriction converts a string to Restriction using the default format. For more details, see the documentation for the Restriction format.

Returns: the Restriction representation of the String

Parameters: - a String

Example Usage: restriction("end before 98-03-25 12:30");

3.11.31 restriction (String, Date)

restriction converts a String and a Date to Restriction using the default format. restriction("end before", date) is equivalent to restriction("end before" & date.string). For more details, see the documentation for the Restriction format.

Returns: the Restriction representation of the String

Parameters: - a String and a Date

Example Usage: restriction("end before", now + "1 week");

3.11.32 restriction (String, Date_Range)

restriction converts a String and a Date_Range to Restriction using the default format. restriction("end in", period) is equivalent to restriction("end in" & period.string). For more details, see the documentation for the Restriction format.

Basic Functions

satisfies (Restriction, Date_Range)

Returns: the Restriction representation of the String

Parameters: - a String and a Date

Example Usage: `resinjection("in", date_range(now + "1 day", now + "1 week"));`

3.11.33 satisfies (Restriction, Date_Range)

Test to see if a Date_Range satisfies a Restriction.

Returns: Returns TRUE if the Date_Range satisfies the conditions specified by a Restriction.

Parameters: - a Restriction and a Date_Range

Example Usage:

```
do(define(range, date_range(now + "1 day", now + "1 week")),
  satisfies(resinjection("start first in", period), period)); // returns TRUE.
```

3.12 Date_Range Functions

A Date_Range defines a time period using an inclusive starting Date and an exclusive ending Date. The functions below can be used to create, query, and manipulate Date_Ranges.

3.12.1 date_range (Date, Date)

date_range creates a new time period given two dates. The first date must occur before the second date.

Returns: a new time period (Date_Range)

Parameters: - a Date

- a Date; must occur after first Date

Example Usage:

```
date_range("00-1-1 00:00", "00-2-1 00:00");
date_range("00-1-1 00:00", "00-1-1 00:00");
date_range("00-2-1 00:00", "00-1-1 00:00"); // returns an error
```

3.12.2 date_range (Date, Time)

date_range creates a new time period given a date and a time interval. If the time interval is positive, the date is the starting date. If the time interval is negative, the date is the ending date.

Returns: a new time period (Date_Range)

Basic Functions

date_range (String)

Parameters: - a Date
- a Time interval

Example Usage:

```
date_range("00-1-1 00:00", "1 week"); // returns 00-01-01 00:00 <
date_range("00-2-14 00:00", "-1 week"); // returns 00-02-07 00:00 <
// 00-02-14 00:00
```

3.12.3 date_range (String)

date_range converts a string into a time period using the default format. The string must contain two dates with the early date preceding the later date.

Returns: the Date_Range representation of the String

Parameters: - a String; the string must contain two dates with the early one first in the following format: "YY-MM-DD hh:mm YY-MM-DD hh:mm"

Example Usage:

```
date_range("00-1-1 00:00 / 00-1-5 00:00");
date_range("97-9-4 00:00 97-10-1 00:00");
date_range("98-6-1 00:00 / 98-5-3 00:00"); // returns an error
date_range("00-1-1 00:00 / 00:00"); // returns an error
```

3.12.4 date_range (String, Symbol)

date_range converts a string into a time period using the named format. The string must contain two dates. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: the Date_Range representation of the String

Parameters: - a String; the string must contain two dates

- the named format (a Symbol) to use

Example Usage:

3.12.5 enclose (Date, Date_Range)

enclose adjusts a time period to include a date. If the time period already covers the date, then no adjust is made.

Returns: an encompassing time period (Date_Range)

Basic Functions	enclose (Date_Range, Date_Range)
-----------------	------------------------------------

Parameters: - the Date to include in a time period
- the time period (Date_Range) to adjust

Example Usage:

Given...

a = date_range("00-1-1 00:00", "00-3-9 00:00");

Then...

```
enclose("00-4-1 00:00", a); // returns 00-01-01 00:00 / 00-04-01 00:00
enclose("00-2-3 00:00", a); // returns 00-01-01 00:00 / 00-03-09 00:00
enclose("99-11-3 00:00", a); // returns 99-11-03 00:00 / 00-03-09 00:00
```

3.12.6 enclose (Date_Range, Date_Range)

enclose adjusts a time period to include another time period. If the time period already covers the other time period, then no adjustment is made.

Returns: an encompassing time period (Date_Range)

Parameters: - the time period (Date_Range) to cover
- the time period (Date_Range) to adjust

Example Usage:

Given...

```
a = date_range("00-1-1 00:00", "00-3-9 00:00");
b = date_range("99-12-15 00:00", "99-12-25 00:00");
c = date_range("00-5-1 00:00", "00-8-30 00:00");
d = date_range("99-11-1 00:00", "00-1-31 00:00");
```

Then...

```
enclose(b, a); // returns 99-12-15 00:00 / 00-03-09 00:00
enclose(c, a); // returns 00-01-01 00:00 / 00-08-30 00:00
enclose(b, d); // returns 99-11-01 00:00 / 00-01-31 00:00
enclose(d, a); // returns 99-11-01 00:00 / 00-03-09 00:00
```

3.12.7 end (Date_Range)

end either sets or returns the ending date of a time period. When setting, if the end date occurs before the starting date, then the start date is adjusted to match the new ending date.

Basic Functions	intersects (Date_Range, Date_Range)
-----------------	---------------------------------------

Returns: the ending Date of a time period

Parameters: - a time period (Date_Range)

Example Usage:

Given...

a = date_range("00-1-1 00:00:00 / 00-4-15 00:00:00");

Then...

```
end(a); // returns 00-04-15 00:00:00
a.end = "00-2-10 00:00:00"; // changes the range to
// 00-01-01 00:00:00 / 00-02-10 00:00:00
a.end = "99-6-2 00:00:00"; // changes the range to
// 99-06-02 00:00:00 / 99-06-02 00:00:00
```

3.12.8 intersects (Date_Range, Date_Range)

intersects determines if two time periods overlap.

Returns: True if the two time periods overlap, False otherwise.

Parameters: - a time period (Date_Range)
- a time period (Date_Range)

Example Usage:

Given...

```
a = date_range("00-1-1 00:00", "00-3-9 00:00");
b = date_range("99-12-15 00:00", "99-12-25 00:00");
c = date_range("00-5-1 00:00", "00-8-30 00:00");
d = date_range("99-11-1 00:00", "00-1-31 00:00");
```

Then...

```
intersects(b, a); // returns False
intersects(c, a); // returns False
intersects(b, d); // returns True
intersects(d, a); // returns True
```

3.12.9 limit (Date, Date_Range)

limit either returns a date if it is within a time period or the closest boundary of the time period.

Returns: a Date within the time period

Parameters: - a Date

- a time period (Date_Range)

Example Usage:

Given...

```
a = date_range("97-3-14 00:00:00 / 97-5-20 00:00:00");
```

Then...

```
limit("97-4-26 00:00:00", a); // returns 97-04-26 00:00:00
```

```
limit("97-2-1 00:00:00", a); // returns 97-03-14 00:00:00
```

```
limit("97-8-4 00:00:00", a); // returns 97-05-20 00:00:00
```

3.12.10 limit (Date_Range, Date_Range)

limit returns a time period which represents the intersect of two time periods. If the two time periods do not intersect, then the first time period's maximum value is returned as a time period.

Returns: the intersection time period (Date_Range)

Parameters: - a time period (Date_Range)

- a time period (Date_Range)

Example Usage:

Given...

```
a = date_range("00-1-1 00:00", "00-3-9 00:00");
```

```
b = date_range("99-12-15 00:00", "99-12-25 00:00");
```

```
c = date_range("00-5-1 00:00", "00-8-30 00:00");
```

```
d = date_range("99-11-1 00:00", "00-1-31 00:00");
```

Then...

```
limit(b, a); // returns 99-12-25 00:00 / 99-12-25 00:00
```

```
limit(c, a); // returns 00-08-30 00:00 / 00-08-30 00:00
```

```
limit(b, d); // returns 99-11-01 00:00 / 99-12-25 00:00
```

```
limit(d, a); // returns 00-01-01 00:00 / 00-01-31 00:00
```

3.12.11 set_end (Date_Range, Date)

set_end defines the end date of a time period. If the new end date occurs prior to the time period's start date, then the time period's start date is also set the new end date.

Returns: a time period's end Date

Parameters: - a time period (Date_Range)

- a new end Date

Example Usage:

Given...

```
a = date_range("99-12-15 00:00", "99-12-25 00:00");
```

```
b = date_range("00-5-1 00:00", "00-8-30 00:00");
```

Then...

```
set_end(a, "99-12-20 00:00"); // returns 99-12-20 00:00
```

```
set_end(b, "00-9-30 00:00"); // returns 00-09-30 00:00
```

```
set_end(a, "99-11-1 00:00"); // returns 99-11-01 00:00
```

3.12.12 set_start (Date_Range, Date)

set_start defines the start date of a time period. If the new start date occurs after to the time period's end date, then the time period's end date is also set the new start date.

Returns: a time period's start Date

Parameters: - a time period (Date_Range)

- a new start Date

Example Usage:

Given...

```
a = date_range("99-12-15 00:00", "99-12-25 00:00");
```

```
b = date_range("00-5-1 00:00", "00-8-30 00:00");
```

Then...

```
set_start(a, "99-12-20 00:00"); // returns 99-12-20 00:00
```

```
set_start(b, "00-4-1 00:00"); // returns 00-04-01 00:00
```

```
set_start(a, "99-12-28 00:00"); // returns 99-12-28 00:00
```

3.12.13 set_time (Date, Range, Time)

set_time defines a time period's duration, moving the end date as necessary. The duration must be greater than or equal to zero.

Returns: the time period's duration (a Time)

Parameters: - a time period (Date, Range)
- the new duration (Time); must be ≥ 0

Example Usage:

Given...
a = date_range("99-12-15 00:00", "99-12-25 00:00");
b = date_range("00-5-1 00:00", "00-8-30 00:00");

Then...
set_time(a, "7 day"); // returns 7 days; a's end date is changed
// to 199-2-22 00:00
set_time(b, "8 hr"); // returns 8 hr; b's end date is changed
// to 00-05-01 08:00
set_time(a, "-2 day"); // returns an error

3.12.14 start (Date, Range)

start either defines or returns a time period's start date. When defining, if the new start date occurs after to the time period's end date, then the time period's end date is also set the new start date.

Returns: a time period's start Date

Parameters: - a time period (Date, Range)

Example Usage:

Given...
a = date_range("99-12-15 00:00", "99-12-25 00:00");
b = date_range("00-5-1 00:00", "00-8-30 00:00");
Then...
start(a, "99-12-20 00:00"); // returns 99-12-20 00:00
start(b, "00-4-1 00:00"); // returns 00-04-01 00:00

a.start = "99-12-28 00:00"; // returns 99-12-28 00:00

3.12.15 time (Date, Range)

time either defines or returns a time period's duration. When defining, the time period's end date is moved as necessary. Also, the duration must be greater than or equal to zero.

Returns: the time period's duration (a Time)

Parameters: - a time period (Date, Range)

Example Usage:

Given...
a = date_range("99-12-15 00:00", "99-12-25 00:00");
b = date_range("00-5-1 00:00", "00-8-30 00:00");

Then...
time(a, "7 day"); // returns 7 day
time(b, "8 hr"); // returns 8 hr
a.time = "5 day"; // returns 5 day; a's end date is changed to
// 99-12-20 00:00
b.time = "-8 hr"; // returns an error

3.12.16 within (Date, Date, Range)

within determines if a date falls within a time period.

Returns: True if the Date is within the time period; False otherwise.

Parameters: - a Date
- a time period (Date, Range)

Example Usage:

Given...
a = date_range("00-1-1 00:00", "00-3-9 00:00");
Then...
within("00-4-1 00:00", a); // returns False
within("00-2-3 00:00", a); // returns True
within("99-1-1 3 00:00", a); // returns False

3.12.17 within (Date_Range, Date_Range)

within determines if a time period falls completely within another time period.

Returns: True if the first time period falls completely within the second time period;
False otherwise

Parameters: - a time period (Date_Range)
- a time period (Date_Range)

Example Usage:

```
Given...
a = date_range("00-1-1 00:00", "00-3-9 00:00");
b = date_range("99-12-15 00:00", "99-12-25 00:00");
c = date_range("00-5-1 00:00", "00-8-30 00:00");
d = date_range("99-11-1 00:00", "00-1-31 00:00");
```

Then...

```
within(b, a): // returns False
within(c, a): // returns False
within(b, d): // returns True
within(d, a): // returns False
```

3.13 Date_Range_List Functions

Date_Range_List are lists of consecutive time periods (date ranges). The below functions generate consecutive time periods (Date_Ranges), with no gaps, suitable for use as horizons for planning.

3.13.1 days (Date_Range)

days creates a list of consecutive one day time periods which span the given time period. Each time period starts and ends on day boundaries (00:00), except perhaps the start of the first and end of the last day, which will be the start and end time of the given time period.

Returns: A List of consecutive one day Date_Ranges with no gaps, spanning the requested time period.

Parameters: - the time period to span (Date_Range)

Example Usage:

Given...

```
a = date_range("97-06-01 06:00", "97-06-03 14:00");
b = date_range("97-06-10 02:00", "97-06-10 22:00");
c = date_range("97-06-15 00:00", "97-06-17 00:00");
```

Then...

```
days(a): // returns a list containing three time periods:
// { 97-06-01 06:00 / 97-06-02 00:00,
// 97-06-02 00:00 / 97-06-03 00:00,
// 97-06-03 00:00 / 97-06-03 14:00 }
```

```
days(b): // returns a list containing one time period:
// { 97-06-10 02:00 / 97-06-10 22:00 }
```

```
days(c): // returns a list containing two time periods:
// { 97-06-15 00:00 / 97-06-16 00:00,
// 97-06-16 00:00 / 97-06-17 00:00 }
```

3.13.2 days (Date_Range, Time)

days creates a list of consecutive one day time periods which span the given time period. Each time period starts and ends on day boundaries plus the given time offset except perhaps the start of the first and end of the last day, which will be the start and end time of the given time period.

Returns: A List of consecutive one day Date_Ranges with no gaps, spanning the requested time period.

Parameters: - the time period to span (Date_Range)
- the starting Time of each time period; must be ≥ 0
if greater than 24 hours, then first time period will span more than a single day.

Example Usage:

```
Given...
a = date_range("97-06-01 06:00", "97-06-03 14:00");
b = date_range("97-06-10 02:00", "97-06-11 22:00");
c = date_range("97-06-15 00:00", "97-06-20 00:00");
```

Then...

```
days(a, "4 hr"): // returns a list containing three time periods:
```

```
// { 97-06-01 06:00 / 97-06-02 04:00,
// 97-06-02 04:00 / 97-06-03 04:00,
// 97-06-03 04:00 / 97-06-03 14:00 }

days(b, "600 s"); // returns a list containing two time periods:
// { 97-06-10 02:00 / 97-06-11 00:10,
// 97-06-11 00:10 / 97-06-11 22:00 }

days(c, "48 hr"); // returns a list containing three time periods:
// { 97-06-15 00:00 / 97-06-18 00:00,
// 97-06-18 00:00 / 97-06-19 00:00,
// 97-06-19 00:00 / 97-06-20 00:00 }
```

days(a, "-1 hr"); // returns an error

3.13.3 months (Date_Range)

months creates a list of consecutive one month time periods which span the given time period. Each time period starts and ends on month boundaries (first day of the month at 00:00), expect perhaps the start of the first and end of the last month, which will be the start and end time of the given time period.

Returns: A List of consecutive one month Date_Ranges with no gaps, spanning the requested time period.

Parameters: - the time period to span (Date_Range)

Example Usage:

Given...

```
a = date_range("97-06-01 06:00", "97-08-03 14:00");
b = date_range("97-06-10 02:00", "97-06-29 22:00");
c = date_range("97-06-15 00:00", "97-07-17 00:00");
```

Then...

months(a): // returns a list containing three time periods:

```
// { 97-06-01 06:00 / 97-07-01 00:00,
// 97-07-01 00:00 / 97-08-01 00:00,
// 97-08-01 00:00 / 97-08-03 14:00 }
```

months(b): // returns a list containing one time period:

```
// { 97-06-10 02:00 / 97-06-29 22:00 }
```

```
months(c): // returns a list containing two time periods:
// { 97-06-15 00:00 / 97-07-01 00:00,
// 97-07-01 00:00 / 97-07-17 00:00 }
```

3.13.4 quarters (Date_Range)

quarters creates a list of consecutive one quarter time periods which span the given time period. Each time period starts and ends on quarter boundaries (first day of the first, fourth, seventh, or ninth month at 00:00), expect perhaps the start of the first and end of the last quarter, which will be the start and end time of the given time period.

Returns: A List of consecutive one quarter Date_Ranges with no gaps, spanning the requested time period.

Parameters: - the time period to span (Date_Range)

Example Usage:

Given...

```
a = date_range("97-06-01 06:00", "97-10-03 14:00");
b = date_range("97-07-10 02:00", "97-09-01 22:00");
c = date_range("97-06-15 00:00", "97-07-17 00:00");
```

Then...

quarters(a): // returns a list containing three time periods:

```
// { 97-06-01 06:00 / 97-07-01 00:00,
// 97-07-01 00:00 / 97-10-01 00:00,
// 97-10-01 00:00 / 97-10-03 14:00 }
```

quarters(b): // returns a list containing one time period:

```
// { 97-07-10 02:00 / 97-09-01 22:00 }
```

quarters(c): // returns a list containing two time periods:

```
// { 97-06-15 00:00 / 97-07-01 00:00,
// 97-07-01 00:00 / 97-07-17 00:00 }
```

3.13.5 shifts (Date_Range)

shifts creates a list of consecutive one shift time periods which span the given time period. Each time period starts and ends on shift boundaries, expect perhaps the start of the first and end of the last shift, which will be the start and end time of the given time period. Shifts default to 8 hours starting at 00:00.

Returns: A List of consecutive one shift Date_Ranges with no gaps, spanning the requested time period.

Parameters: - the time period to span (Date_Range)

Example Usage:

Given...

```
a = date_range("97-06-01 06:00", "97-06-01 23:00");
b = date_range("97-06-10 09:00", "97-06-10 15:00");
c = date_range("97-06-15 00:00", "97-06-15 12:00");
```

Then...

shifts(a): // returns a list containing three time periods:

```
// { 97-06-01 06:00 / 97-06-01 08:00,
// 97-06-01 08:00 / 97-06-01 16:00,
// 97-06-01 16:00 / 97-06-01 23:00 }
```

shifts(b): // returns a list containing one time period:

```
// { 97-06-10 09:00 / 97-06-10 15:00 }
```

shifts(c): // returns a list containing two time periods:

```
// { 97-06-15 00:00 / 97-06-15 08:00,
// 97-06-15 08:00 / 97-06-15 12:00 }
```

3.13.6 shifts (Date_Range, Time)

shifts creates a list of consecutive one shift time periods which span the given time period. Each time period starts and ends on shift boundaries plus the given time offset, except perhaps the start of the first and end of the last day, which will be the start and end time of the given time period.

Returns: A List of consecutive one day Date_Ranges with no gaps, spanning the requested time period.

Parameters: - the time period to span (Date_Range)
- the length of the shift (Time): must be > 0

Example Usage:

Given...

```
a = date_range("97-06-01 06:00", "97-06-01 15:00");
b = date_range("97-06-10 09:00", "97-06-10 09:18");
c = date_range("97-06-15 00:00", "97-06-16 17:00");
```

Then...

shifts(a, "4 hr"): // returns a list containing three time periods:

```
// { 97-06-01 06:00 / 97-06-01 08:00,
// 97-06-01 08:00 / 97-06-01 12:00,
// 97-06-01 12:00 / 97-06-01 15:00 }
```

shifts(b, "600 s"): // returns a list containing two time periods:

```
// { 97-06-10 09:00 / 97-06-10 09:10,
// 97-06-10 09:10 / 97-06-10 09:18 }
```

shifts(c, "14 hr"): // returns a list containing three time periods:

```
// { 97-06-15 00:00 / 97-06-15 14:00,
// 97-06-15 14:00 / 97-06-16 00:00,
// 97-06-16 00:00 / 97-06-16 14:00 }
// 97-06-16 14:00 / 97-06-16 17:00 }
```

shifts(a, "0 hr"): // returns an error

shifts(a, "-1 hr"): // returns an error

3.13.7 shifts (Date_Range, Time, Time)

shifts creates a list of consecutive one shift time periods which span the given time period. Each time period starts and ends on shift boundaries plus the given time offset, except perhaps the start of the first and end of the last day, which will be the start and end time of the given time period. The length of a shift is defined by the first time.

Returns: A List of consecutive one day Date_Ranges with no gaps, spanning the requested time period.

Parameters: - the time period to span (Date_Range)
- length of the shift (Time): must be > 0
- the starting Time of each time period: must be >= 0;
if greater than the shift length, then first time period
will span more than a single shift

Example Usage:

Given...

```
a = date_range("97-06-01 06:00", "97-06-01 16:00");
b = date_range("97-06-10 09:00", "97-06-10 17:00");
c = date_range("97-06-15 00:00", "97-06-15 17:00");
```

Then...

```
shifts(a, "4 hr", "2 hr"); // returns a list containing three time periods:
// { 97-06-01 06:00 / 97-06-01 10:00,
// 97-06-01 10:00 / 97-06-01 14:00,
// 97-06-01 14:00 / 97-06-01 16:00 }
```

```
shifts(b, "8 hr", "600 s"); // returns a list containing two time periods:
// { 97-06-10 09:00 / 97-06-10 16:10,
// 97-06-10 16:10 / 97-06-10 17:00 }
```

```
shifts(c, "4 hr", "10 hr"); // returns a list containing three time periods:
// { 97-06-15 00:00 / 97-06-15 10:00,
// 97-06-15 10:00 / 97-06-15 14:00,
// 97-06-15 14:00 / 97-06-15 17:00 }
```

```
shifts(a, "0 hr", "2 hr"); // returns an error
```

```
shifts(a, "-1 hr", "0 hr"); // returns an error
```

3.13.8 weeks (Date, Range)

weeks creates a list of consecutive one week time periods which span the given time period. Each time period starts and ends on week boundaries (Monday at 00:00), except perhaps the start of the first and end of the last month, which will be the start and end time of the given time period.

Returns: A List of consecutive one week Date_Ranges with no gaps, spanning the requested time period.

Parameters: - the time period to span (Date, Range)

Example Usage:

Given...

```
a = date_range("97-06-04 06:00", "97-06-18 14:00");
b = date_range("97-06-10 02:00", "97-06-14 22:00");
c = date_range("97-06-15 00:00", "97-06-20 00:00");
```

Then...

```
weeks(a); // returns a list containing three time periods:
// { 97-06-04 06:00 / 97-06-09 00:00,
// 97-06-09 00:00 / 97-06-16 00:00,
// 97-06-16 00:00 / 97-06-18 14:00 }
```

```
weeks(b); // returns a list containing one time period:
// { 97-06-10 02:00 / 97-06-14 22:00 }
```

```
weeks(c); // returns a list containing two time periods:
// { 97-06-15 00:00 / 97-06-16 00:00,
// 97-06-16 00:00 / 97-06-20 00:00 }
```

3.13.9 weeks (Date, Range, Integer)

weeks creates a list of consecutive one week time periods which span the given time period. Each time period starts and ends on week boundaries plus the given day offset except perhaps the start of the first and end of the last day, which will be the start and end time of the given time period.

Returns: A List of consecutive one week Date_Ranges with no gaps, spanning the requested time period.

Parameters: - the time period to span (Date, Range)

- the day offset (0 = Sunday, 1 = Monday, etc.); must be >= 0 and <= 6

Example Usage:

Given...

```
a = date_range("97-06-04 06:00", "97-06-18 14:00");
b = date_range("97-06-07 02:00", "97-06-07 22:00");
c = date_range("97-06-15 00:00", "97-06-20 00:00");
```

Then...

```
weeks(a, 2); // returns a list containing three time periods:
// { 97-06-04 06:00 / 97-06-10 00:00,
// 97-06-10 00:00 / 97-06-17 00:00,
// 97-06-17 00:00 / 97-06-18 14:00 }
```

```
weeks(b, 0); // returns a list containing one time period:
```

```
// { 97-06-07 02:00 / 97-06-07 22:00 }

weeks(c, 4): // returns a list containing two time periods:
// { 97-06-15 00:00 / 97-06-19 00:00,
// 97-06-19 00:00 / 97-06-20 00:00 }
```

3.14 Logical Functions

Logical Functions evaluate logical expressions and return either true or false. They are useful for decision making.

3.14.1 and (Logical, Logical)

and returns "true" if all of the given logical parameters are "true". If no parameters are given, and returns "true". Evaluation is short circuited. In other words, parameters are evaluated in the order listed. If the evaluation of parameter returns "false", and immediately exits, returning "false" without evaluating the remaining parameters.

Returns: TRUE if all parameters evaluate to "true";
FALSE otherwise

Parameters: - Any number of Logical expressions.

Example Usage:

```
Given...
a = 5
b = 0
```

Then...

```
and(a > 3, a <= 9) // Returns true
and(a > 3, a <= 9, b == 0) // Returns true
and(a > 6, b == 0) // Returns false
```

3.14.2 not (Logical)

not computes the negation of its logical parameter. not(b) is equivalent to !b.

Returns: TRUE if the parameter evaluates to "false";
FALSE if the parameter evaluates to "true"

Parameters: - a Logical expression

Example Usage:

```
not(5 > 3) // returns false
not(0 == 1) // returns true
```

3.14.3 or (Logical, Logical)

or returns "true" if any of the given logical parameters are "true". All parameters are evaluated. If no parameters are given, or returns "false".

Returns: TRUE if any parameter evaluates to "true";
FALSE if all parameters evaluate to "false"

Parameters: - Any number of logical expressions.

Example Usage:

```
Given...
a = 5
b = 1
```

Then...

```
or(a < 3, a > 9) // Returns false
or(a < 3, a > 9, b != 0) // Returns true
```

3.14.4 xor (Logical, Logical)

xor computes the sequential exclusive or of all the logical parameters. In other words, xor is applied to the first two parameters. The result becomes the first parameter to the next two value xor evaluation, etc.

The exclusive or of two values is "true" if one of the values is "false" while the other is "true". The result is "false" if both values are "true" or both values are "false".

xor(a, b) is the same as a != b;

Returns: TRUE if the sequential exclusive or of all parameters do not match;
FALSE otherwise.

Parameters: - Any number of logical expressions.

Example Usage:

```
Given...
a = 5
```

Basic Functions	Miscellaneous Functions
-----------------	-------------------------

```

b = 1

Then....
xor(a < 3, a > 9) // returns true
xor(a < 3, a > 9, b != 0) // returns false

```

3.15 Miscellaneous Functions

3.15.1 id (Void)

Convert any Model instance to an ID. Strings can be converted too; the format is "0x" followed by a hexadecimal number. For example: id("0x12345678")

Returns: an ID

Parameters: Any Model, or a string

Example Usage:

```

@~op_plan.id == op_plan.id; // is true.
op_plan.id.string.id == op_plan.id; // is true.

```

ID's can be used to uniquely identify models that don't have unique key-fields (an Operation_Plan, for example). You can make an ID out of any model, convert it to string (using the string`OIL` function which converts anything to a string), send the string to some 3rd party software package, send it back, convert it back to an ID (using this id(string) OIL function) then search for the model comparing ID objects (which is much faster than comparing strings)

WARNING: Id strings are usually the hex address of the object in memory. Therefore they are only valid for the current invocation of the engine. This is usually good enough for the user-interface uses 'id' was designed for.

3.15.2 current_index (Cell)

Returns the current index of a replicating cell

Returns:

Parameters:

Example Usage:

Basic Functions	system (String)
-----------------	-----------------

3.15.3 system (String)

Passes 'string' to the operating system for execution (On UNIX, it uses the 'sh' shell).

Returns: If the operating system returns a code from the shell, that code is returned; otherwise 0 is returned. (For UNIX systems, non-zero return values indicate the execution failed or had some sort of error.)

Parameters: The shell command(s) to execute.

Example Usage:

```

system("if [ -f $12_DATA/AM_backup/scp.i2 ]; then"
"find/mv $12_DATA/AM_backup/scp.i2 $12_DATA/AM_backup/scp.i2.bak; fi");

```

3.15.4 values (Type)

If the parameter is an enumeration type, return the list of valid values. (model extension selectors have enumeration types named "cmodel_name>_selector_field_name>_Enum" Example: values(Operation_process_Enum)

Returns:

Parameters:

Example Usage:

3.15.5 data (Symbol)

This function is obsolete. Find or Create a user-defined DataBase model whose name is given by the parameter string. (see the documentation for the "DataBase" model).

Returns:

Parameters:

Example Usage:

3.15.6 exists (Void)

Test for nonexistent values. If the parameter returns an error, 'exists' will return the same error. See 'exists_without_errors'

Returns: TRUE if it's parameter exists (is not nonexistent).

Parameters: One parameter of any type.

Example Usage: if (exists(model.field, model.field.string, ""));
Note: this example would be better coded as: model.field.string ? "" :

3.15.7 exists_without_errors (Void)

Test for nonexistent and error values. Throws away any error messages.

Returns: TRUE if it's parameter exists (is not nonexistent), and is not an error.

Parameters: One parameter of any type.

Example Usage: exists_without_errors(date("bad date")); // returns FALSE.

3.15.8 export (string, void)

Writes data to a file or directory of files.

If the first parameter is a directory, and there is a before_export.in file in the directory, then the OIL commands in that file will be evaluated (see do_file) before the files in the directory are exported. Similarly if the directory contains an after_export.in file, then those OIL commands will be evaluated.

The %DATA% environment variable is bound to the export directory while evaluating the before_export.in, after_export.in and the export worksheets.

Returns: True on success, false on errors.

Parameters: - (pathname [, optional parameters])

The first parameter specifies where to read the export files.

- If it names a directory on the include path, then all the *exp files in that directory are read and processed.
- If it names a file on the include path, then that specific file read and processed.

@- The rest of the parameters are passed to the export worksheet.

Example Usage:

export(directory/file, operation_plans);

3.15.9 import (string, void)

Reads data from a file or directory of files.

If the first parameter is a directory, and there is a before_import.in file in the directory, then the OIL commands in that file will be evaluated (see do_file) before the files in the directory are imported. Similarly if the directory contains an after_import.in file, then those OIL commands will be evaluated.

The %DATA% environment variable is bound to the import directory while evaluating the before_import.in, after_import.in and the import worksheets.

Returns: True on success, false on errors.

Parameters: - (pathname [, optional parameters])

The first parameter specifies where to read the data files.

- If it names a directory on the include path, then all the *imp files in that directory are read and processed.
- If it names a file on the include path, then that specific file read and processed.

The rest of the parameters are passed to the import worksheet.

Example Usage:

import(data/orders, seller, site);

3.15.10 engine_name ()

Return the base name of the last saved/opened model file. If there is none, then return the base name of the directory specified by the data option.

Parameters:

Example Usage:

3.15.11 get_color (String)

Invoke the color chooser.

Returns: the color selected in the color chooser.

Parameters: the color of choice.

Example Usage: get_color(blue)

3.15.12 get_drag_string (Integer)

Return the nth drag String argument.

Parameters: an integer indicating which drag string argument to return.

Example Usage:

3.15.13 getenv (String)

Get the value of the operating system environment variable with the specified name String.

Returns: the current setting of the environment variable

Parameters: - an environment variable name

Example Usage: `getenv("I2_IMPORT")` //returns the current value of this //environment variable

3.15.14 translate (String)

Translate the argument to users desired language.

Returns: String

Parameters: String

Example Usage: `[A1,title = translate("Promise As Planned");]`

3.15.15 model_file ()

Return the full path name of the last saved/opened model file.

Parameters: None.

Example Usage:

3.15.16 option (String)

Returns the value of the named option

Parameters: string name of an option

Example Usage:

3.15.17 setenv (String, String)

Sets the value of the operating system environment variable named by the first String to the value specified by the second String.

Returns: Returns the value String.

Parameters: - a String (name of variable) - a String (value for variable)

Example Usage:

`setenv("I2_EXPORT", "users/me")` //sets the value of //I2_EXPORT to the directory /users/me

3.15.18 typed_value (void)

Convert anything into a Typed_Value. Typed_Value is a typed/value pair, which allows us to do run-time typing.

If passed a Cell_State, the typed_value will be the contents of the worksheet cell pointed to by the Cell_State.

Returns: a Typed_Value

Parameters: - One parameter of any type.

Example Usage:

```
@- define(iv, typed_value("hi there"));
iv.type: -> String
iv.type == "string": -> TRUE
iv.value(string): -> "hi there"
iv.string: -> "hi there" // string converts *anything* to a string
```

We can use this to create lists of lists:

```
define(iv_list, list(typed_value(list("one", "two")),
typed_value(list("three", "four", "five"))));
iv_list.first.type: -> list(Typed_Value);
iv_list.for_each(#.value(list(string))) = one, two, three, four, five
```

As a corollary, we can use this to hide lists inside an axis/cross replicator.

Comparing a type to a constant string is very fast, because the

string is automatically converted to a Value_Type during expression compilation. The comparison is a trivial pointer match. Strings of if's for doing different things with different types should be fast. For example:

```
if(iv.type == "Location", iv.value(Location),  
if(iv.type == "Buffer", iv.value(Buffer).location,  
if(iv.type == "Resource", iv.value(Resource).location)));
```

3.15.19 owner (Typed_Value)

Return the owner of a model as a Typed_Value.

Parameters:

Example Usage:

3.15.20 first_value (Typed_Value)

If the Typed_Value passed in is a List type, return the first element of the list as a Typed_Value; otherwise return NONEXISTENT. If the list is empty, return Typed_Value(clement_type, NONEXISTENT)

Returns: A Typed_Value

Parameters: A Typed_Value

Example Usage:

3.15.21 delete (Void)

Eliminate the model object passed in the parameter. Use with caution.

Returns: Nothing.

Parameters:

Example Usage:

3.15.22 make_type (Void, Type)

make_type returns a value with a specified type.

Returns: The first parameter converted to the type specified by the 2nd parameter.

Parameters: - a value - a type

Example Usage: normal db (Buffer_Plan bp) { variable dd = make_type(nonexistent, List(Sic)); ... {f? = dd.first(); //won't display until the List has at least one member ... }

3.15.23 nonexistent()

Returns the nonexistent value (or lack of a value). This value will cause most other functions to abort.

Parameters:

Example Usage:

3.15.24 set_option (String, String)

Sets the value of the named option

Returns: Nothing.

Parameters:

- name of a command line option.
- value for the named command line option.

Example Usage: set_option("help", "all"); // Prints the list of all options

3.15.25 type (Typed_Value)

Return the type contained in a Typed_Value.

Parameters: Typed_Value.

Example Usage:

3.15.26 type (String)

Converts a string to a Type

Returns: The named type or MODEL_ERROR

Parameters: A string naming the Type to be returned.

Example Usage: type("integer")

3.15.27 model_type (void)

If the parameter is a model instance, return it's Model_Type.
If the parameter is a Typed_Value, and the Typed_Value contains a model, return the Model_Type contained by the Typed_Value.
Otherwise, return NONEXISTENT.

Returns:

Parameters:

Example Usage:

3.15.28 value (Typed_Value, Type)

Return the value member of a Typed_Value cast to the named type.

The second argument is a constant string which names the OIL type, like Resource_Plan. The function checks the correctness of the conversion, and returns NONEXISTENT if it's wrong.

This function can be used with the model_choose function to enhance the model_choose reports.

Parameters:

- a Typed_Value.
- a constant string which names the OIL type, like Resource_Plan.

Example Usage:

```
compute chosen_skill = value(typed_value(cell_state), "Resource_Plan");  
compute loc = iv.value("Location") ?  
    iv.value("Buffer").location ?  
    iv.value("Resource").location);
```

3.15.29 trace_time ()

The trace_time function can take any number of parameters that return any Type. It will evaluate each of the parameters in order, and return the result of the last one. It also prints how long it took to evaluate the parameters.

Example Usage:

```
$S for each user: 244260ms system: 108460ms cpu: 352720ms real 715926ms  
trace_time(  
    supply_chains.find("TOSHIBA"),  
    plans.find("TOSHIBA"),  
    seller_plans.for_each(# product_forecasts.
```

```
    for_each(#.entries.for_each(#.accept_as_allocated)))  
)
```

Output:

```
user - CPU milliseconds used to execute the parameters to trace_time  
system - CPU milliseconds used by the system.  
cpu - Total CPU milliseconds (user + system)  
real - Actual (wall-clock) milliseconds (includes time used by other  
        processes which are running).
```

Returns: The result from the last parameter.

Parameters: - Any.

3.15.30 memory_size ()

Returns the memory actually used by the process which is running the memory_size function. The memory_size changes as memory is allocated and freed from the process (see the process_size function). Therefore memory_size may decrease even though process_size stays the same. The difference between the process_size and the memory_size is held in reserve for future use. It's typically a megabyte, but could be much more.

Note: this function does not currently work on Microsoft operating systems.

Returns: memory used in bytes.

Parameters: - none.

Example Usages:

```
echo("Memory size: " & string(memory_size / 1024) & "KB");  
echo("Memory size: " & string(memory_size / (1024 * 1024)) & "MB");  
echo("Memory size: " & string(memory_size / (1024 * 1024 * 1024)) & "GB");
```

3.15.31 process_size ()

Returns the memory used by the process which is running the process_size function. The process_size will never decrease. The memory allocation sub-system will get large chunks of memory from the system (typically 512KB at a time). It then hands out this memory for various purposes. The memory_size function will give a more accurate reading of how much memory is actually in use. process_size is sort-of the max value of memory_size, plus some overhead.

Basic Functions	with_connection (Connection, void)
-----------------	--------------------------------------

Note: Many system commands report memory usage in "kilobytes" by taking total memory size and dividing by 1024 (NOT 1000). If the results from process_size do not match what's reported by your favorite system command, this could be the cause. Note: some system commands (e.g. ps) do not include shared memory, or memory that's swapped out.

Note: this function does not currently work on Microsoft operating systems.

Returns: process size in bytes.

Parameters: - none.

Example Usages:

```
echo("Process size: " & string(process_size / 1024) & "KB");
echo("Process size: " & string(process_size / (1024 * 1024)) & "MB");
echo("Process size: " & string(process_size / (1024 * 1024 * 1024)) & "GB");
```

3.15.32 with_connection (Connection, void)

Like do, except the first parameter is a Connection. All other arguments are executed in the context (with a Cell_State) of the Application_Report for the connection.

Note: Many system commands report memory usage in "kilobytes" by taking total memory size and dividing by 1024 (NOT 1000).

Returns: The result from the last parameter.

Parameters: - A 'Connection' model.

Example Usage:

```
with_connection(users.find("somebody"),active_ui,
display_report("report_name", parameters));
```

3.15.33 functions (Symbol)

Get the list of Functions whose name matches the parameter string. Since there can be lots of functions with the same name (but different parameters), the Function model can't have a 'find' method. This function is used instead. This function is *much* faster than filtering the list of all functions.

Returns: The list of Functions whose name matches the parameter string.

Parameters: a function name

Basic Functions	echo (Logical)
-----------------	------------------

Example Usage:
// Get the documentation for the 'enclose' function that takes a 'Date_Range' parameter
functions("enclose").filter(#argument_types.first == "Date_Range").first.documentation;

3.15.34 echo (Logical)

In the do_file function, expressions are echoed by default. echo(false); will turn off echoing. echo(true); will turn it back on.

3.15.35 echo (String)

Print a constant string to "stderr".

Returns: Nothing.

Parameters:

- a string to print to stderr.

Example Usage:

3.15.36 with_echo_file (String, void)

Similar to do, except the first parameter is a filename. All the other parameters will be evaluated with their output going to the specified file.

Returns: The value of the last parameter

Parameters: - The first parameter is a filename, the other parameters can be anything.

Example Usage:

```
with_echo_file("filename", echo("Hello World"));
```

3.16 File Functions

File functions

3.16.1 is_directory (String)

Returns true if the first parameter is the pathname to a directory.

Returns: True if the first parameter is the pathname to a directory.

Parameters: - A pathname string

Example Usage:
if (is_directory("something"), import("something"));

3.16.2 is_file (String)

Returns true if the first parameter is an existing filename.

Returns: True if the first parameter is an existing filename.

Parameters: - A pathname string

Example Usage:
if (is_file("something"), import("something"));

3.16.3 is_writable (String)

Returns true if the first parameter is writable file or directory.

Returns: True if the first parameter is writable file or directory.

Parameters: - A pathname string

Example Usage:
if (is_writable("something"), export("something"));

3.17 Evaluation Functions

Functions for controlling the evaluation of Expressions.

3.17.1 background (Expression)

The background function can take any number of parameters that return any Type. It will evaluate each of the parameters in order, and return the result of the last one. The parameter expressions may be executed in the background (some of the planning functions will do this). No reply will be sent back to the UI.

Returns: The result from the last expression.

Parameters: - Any

Example Usage:
plan_site_plans_for_each(#promise_as_planned).background;
background(active_strategy.run, wait, update_report);

3.17.2 do ()

The 'do' function can take any number of parameters that return any Type. It will evaluate each of the parameters in order, and return the result of the last one. One of the parameters can be a 'define' function, which defines variables local to the 'do'.

Returns: The last parameter.

Parameters: - Any

Example Usage:
do(define(hi, "Greetings"), echo(hi), echo("From " & username));

3.17.3 call (Symbol)

Evaluate the function worksheet specified by the first parameter, passing parameters specified by the rest of the parameters.

Parameters:

The first parameter is the name of the function worksheet. This must be string constant or identifier. The current user's report path is searched for a file with this name and a ".fws" (functional worksheet) extension.

The rest of the parameters are passed to the function worksheet. If a variable is passed as a parameter, the called function worksheet can change the value of the variable by setting the parameter (e.g. the variable is "passed by reference").

Returns:

If the named function worksheet has a cell named 'return', then the contents of that cell are returned; otherwise 'call' returns void.

Example Usage:
define(result2, make_type(nonexistent, string));
define(result, call(my_oil_function, result2, 123));

3.17.4 do_file_echo (String)

Evaluates all of the expressions in the specified file. The expression and the results are sent to stdout, unless the first character in the file is '@', or 'echo(oF)'; is used. (remnant of old mdsos batch files..)

Parameters: The name of the file to read OIL expressions from. If the file name is "", OIL commands are read from stdin. This can be a useful debugging tool.

Example Usage: `do_file("$12_HOME/custom/my_commands.in");`

3.17.5 do_file (String)

Like `do_file_echo`, except it will optimize itself into a simple `do` if the filename parameter is a string constant, and `echo` is turned off at the time `do_file` is compiled. That is, if it's compiled inside another `do_file` where `echo(off)` is used, or it's invoked as `@do_file("some_file.in")`

Warning: `do_file` will be changed to never echo statements in 3.07. Change your scripts to use `do_file_echo` when you want echoing (typically only regression tests need echoing, and all other production uses don't).

Returns:

If in-lined, The results from the last expression in the file.
Otherwise, it returns void (nothing).

Parameters: The name of the file to read OIL expressions from. If the file name is "", `do_file_echo` is invoked to read OIL commands from `stdin`. This can be a useful debugging tool.

Example Usage: `do_file("$12_HOME/custom/my_commands.in");`

3.17.6 do_file_fast (String)

Like `do_file_echo`, except it will optimize itself into a simple `do`. The filename parameter must be string constant.

Warning: This function is obsolete in 3.06 (replaced by `do_file`).

Parameters: The name of the file to read OIL expressions from.

Returns: The results from the last expression in the file.

Example Usage: `do_file_fast("$12_HOME/custom/my_commands.in");`

3.17.7 engine (Expression)

Send the expression to the engine for evaluation. This is primarily used in Command and Control Expressions that would normally execute in the UI or API clients.

Parameters:

- expression that is to be sent to the engine for evaluation.

Example Usage:

3.17.8 evaluate (Expression)

Compiles and executes the OIL expression passed as a parameter. The expression will be evaluated in the current context, so it can use any local variables, cells, etc.

Parameters: The string to evaluate

Returns: TRUE if there were no compile error, else FALSE

Example Usage: `evaluate(model.user_field); // user_field is a string.`

3.17.9 if (Logical, Void, Void)

The `if` function takes two or three parameters. The first must return Logical. If the first returns "true", then the second parameter is evaluated and its result returned; otherwise, the third parameter (when present) is evaluated and its result returned. Note that if there are 3 parameters, the second and third must return the same Type, and that will be the Type returned by the `if` function. If there are two parameters, `if` returns the "Void" type.

Note that only one of the second or third parameters is evaluated; the other is not evaluated at all. This means that the Logical could be checking for an error condition in one of the other parameters so that the error can be avoided. For example, `if(denom != 0, num/denom, 0)` prevents division by zero errors by checking the denominator and not performing the division if it is 0.

Parameters:

Example Usage:

3.17.10 while (Logical, Void)

Looping function; loop while the first parameter returns true.

WARNING: Most `while` loops can be coded much more efficiently using the `for_catch` function. Any `while` loop that appends to a list should probably be re-coded to use `for_catch`.

Parameters:

The first parameter is an expression that returns FALSE when the loop should exit. while(true) loops forever until the number of iterations specified by the while_max option is exceeded (the default is 4,000,000). The rest of the parameters are simply executed.

Returns: nothing.

Example Usage:

```
define(count, 3);
while(count > 0, echo(count,string), define(count, count - 1));
// Note: it would be much better to use:
// integers(1, 3).for_each(# string,echo);
```

3.17.11 reference (Variable)

Returns a Reference to a variable. A reference can be used with the typed_value function to get the current value of the variable, and with the sel_cell function to set the variable.

Returns: A Reference to the variable

Parameters: A variable

Example Usage:

```
variable foo = "Howdy"; // type is String'
variable foo_reference = reference(foo); // type is Reference'
foo_reference.typed_value.value("string"); // equals "Howdy"
sel_cell(foo_reference.typed_value("Hi")); // Sets 'foo' to "Hi"
```

3.18 RhythmLink Functions

Functions relating to RhythmLink

3.18.1 r_l_field (Data_Table, String)

This function is used in a rhythmLink import layout. The current record is controlled by the traversal mechanism of the import layout.

Returns: current field value belonging to a specific field from the current record of a table.

Parameters:

- first parameter represents the table.
- second parameter represents the field name.

Example Usage:

3.18.2 rhythmLink_field (Symbol, String)

This function returns the named field of a record. It is used internally by a Data_Copy_Map.

Returns: a Value Type

Parameters: - transaction id, field name

Example Usage:

3.18.3 r_l_record_field (Record, String)

This function returns the named field of a record.

Returns: a Value Type

Parameters:

- record
- field name

Example Usage:

3.19 Debug Functions

Functions for OIL debugging

A simple lily debugger is provided with the following OIL functions (optional parameters are in square brackets)

```
@- break([expression [, name [, condition]])
breakpoint([, name])
stop(cell_name [, worksheet_name])
enable(breakpoint_name)
disable(breakpoint_name)
next([count])
step([count])
cont([count])
watch([expression [, name [, condition]])
unwatch(watchpoint_name)
where
whereis
print_variables
```

Basic Functions	stop (Symbol, Symbol)
-----------------	-------------------------

print_report_variables
inspect(any_model)

When stopped at a breakpoint, an "OIL listener" will be entered, which reads from scp_engine's standard-in, and writes to standard-out. You will see a "ODB> " prompt (OIL Debugger). You can enter any OIL expression, and see the evaluation results. Entering an empty line will cause the previous command to be run again. This is handy for running 'hex' or 'cont' repeatedly.

Note: The debugger prompt can be customized via the 'debug_prompt' option. For example:

```
set_option('debug_prompt', "Break (0)>")  
will use the breakpoint name in the prompt (e.g. "Break 3> ")
```

Warning: This means that you can't run scp_engine in the background and still use these debugging aids!

Suggestion: Run scp_engine in an emacs shell buffer. That way you can use meta-p to recall previous commands, you get infinite history, and it's easy to kill&yank (or cut&paste).

NOTE: All breakpoints default to disabled if the 'debug' option is FALSE (that's the default!). Breakpoints default to enabled when the debug option is TRUE. Breakpoint's can be controlled through the Breakpoint model. See the 'enable' and 'disable' OIL functions.

3.19.1 stop (Symbol, Symbol)

Creates a Breakpoint for the named cell and worksheet. See the 'break' function for more details.

3.19.2 stop (Symbol)

Creates a Breakpoint for the named cell in the current worksheet. See the 'break' function for more details.

3.19.3 break (Expression, Symbol, Expression)

Executes the first parameter, returning it's value. Also creates a Breakpoint model for the expression.

Parameters:

If no parameters are specified, the current list of breakpoints is printed.

1st parm can be of any type, and is the result from this function.

Basic Functions	breakpoint ()
-----------------	----------------

2nd parm is the optional breakpoint name (the default is a small integer). Note: There can be multiple breakpoints with the same name. All will be enabled & disabled together, have the same condition, etc.

3rd parm is an optional breakpoint condition which must evaluate to TRUE before breaking. 'condition' is evaluated before 'expression'.

3.19.4 breakpoint ()

Create a breakpoint

3.19.5 breakpoint (Symbol)

Find or create a breakpoint named 'name'

3.19.6 disable ()

Disable all breakpoints

3.19.7 disable (Symbol)

Disable the named breakpoint. If 'name' is "all", then all breakpoints are disabled.

3.19.8 enable ()

Enable all breakpoints

3.19.9 enable (Symbol)

Enable the named breakpoint. If 'name' is "all", then all breakpoints are enabled.

3.19.10 next ()

Run until the next cell, do_file line or variable assignment. Does not cross into other 'call' or 'do_file' files (see step).

3.19.11 step ()

Run until the next cell, do_file line or variable assignment. Will step into 'call' or 'do_file' statements (see next).

3.19.12 cont ()

Continue until the next breakpoint.

3.19.13 next (Integer)

Run until the next cell, do_file line or variable assignment. Does not cross into other 'call' or 'do_file' files (see step). 'count' is the number of breakpoints to skip before actually stopping.

3.19.14 **step (Integer)**

Run until the next cell, do_file line or variable assignment. Will step into 'call' or 'do_file' statements (see next) 'count' is the number of breakpoints to skip before actually stopping.

3.19.15 **cont (Integer)**

Skip 'count' minus one breakpoints, then stop.

3.19.16 **watch (Expression, Symbol, Expression)**

Creates a 'watchpoint' model object that will execute 'expression' and print the results at every breakpoint executed in the same context as 'watch' was executed in (the same worksheet or do_file).

Parameters:

If no parameters are specified, the current list of watchpoints is printed.

1st parm can be of any type, and is the result from this function.

2nd parm is the optional watchpoint name (the default is a small integer). Note: If there is already a watchpoint with the given name, it's expression and condition are replaced with new definitions.

3rd parm is an optional watchpoint condition which must evaluate to TRUE before printing. 'condition' is evaluated before 'stuff'.

If the 3rd 'condition' parameter is nonexistent (the default) the watchpoint will print only when the results are different then the last time. To print every time, use TRUE for the condition.

3.19.17 **unwatch (Symbol)**

Removes the named 'watchpoint' model object (created by 'watchpoint')

Parameters: The name of the watchpoint to delete. If the name is 'all', then all watchpoints are deleted.

3.19.18 **where ()**

Return a one-line description of where this function is executing.

3.19.19 **whereis ()**

Print a detailed description of where this function is executing

3.19.20 **print_variables ()****3.19.21** **print_report_variables ()**

Print all the report variables and their values.

3.19.22 **inspect (void)**

Inspect any model, displaying all it's fields and sub-models. You can "drill down" into the model using the inspectS, inspectH and inspectHS functions.

Parameters: Any model, or list[model]

Example: supply_chains.inspect; // results in:
1 = Inspection of Supply_Chain:

```
==> name          schain
==> description    A simple supply chain schain
0 ==> sites         A-SUPL, A-LINK
1 ==> top_sites     A-LINK, A-SUPL
2 ==> sellers        LINK-SEL, LINK-SUB-SEL2, LINK-SUB-SEL1, SUPL-SEL
3 ==> top_sellers   SUPL-SEL, LINK-SEL
4 ==> plans          plan1
```

3.19.23 **inspectS (Integer)**

Drill down into a field displayed by the previous 'inspect*' call.

Parameters: 'inspect' numbers the fields it prints. Pass one of these numbers into inspectS.

3.19.24 **inspectH (Integer)**

Recall a previously inspected value.

Parameters: Each time one of the 'inspect*' functions is called, the first line shows a history number. Using that number as the parameter to inspectH prints the same inspection again.

3.19.25 **inspectHS (Integer, Integer)**

Drill down into a field of a previously inspected value.

Parameters:

History_number: Each time one of the 'inspect*' functions is called, the first line shows a history number.

@ ~ field_number: 'inspect' numbers the fields it prints. Pass one of these numbers into inspectV.

3.19.26 inspectV (Integer)

Return the value of a field from the last inspected model (or list).

Parameters: 'inspect' numbers the fields it prints. Pass one of these numbers into inspectV.

3.19.27 inspectV (Integer, Integer)

Return the value of a field from the last inspected model (or list).

Parameters:
History_number: Each time one of the 'inspect'* functions is called, the first line shows a history number.

@ ~ field_number: 'inspect' numbers the fields it prints. Pass one of these numbers into inspectV

3.20 Program Functions

Functions for program control

3.20.1 user ()

Returns the User model for the client running this function. If you are executing an OIL expression where there is not a user (for example, the startup expression), then the user OIL function returns the model for the 'unspecified' user.

Parameters: None.

Example Usage:

3.20.2 quit (String)

Quit this client; notify the engine accordingly. The parameter String explains why.

Returns: Nothing.

Parameters:

- string explaining why this client is quitting.

Example Usage:

3.20.3 shutdown (String)

Ask engine to shutdown. The parameter String explains why.

Returns: Nothing.

Parameters:
- string explaining why the engine is being shut down.

Example Usage:

3.21 Engine Queue Functions

Engine_Queue Commands

3.21.1 wait ()

Cause future engine commands to wait until all previous commands have finished.

'wait' only affects the engine<->gui communications mechanisms. You can use it (only) on the gui to serialize engine requests with different priorities.

In general, there are two classes of commands, those that only display data (e.g. display_report, update_report), and those that modify data (e.g. setting a field in a model). The commands that only display data have a higher-priority than the modify commands, so they normally run first. 'wait' can be used to change that.

For example, you may have an action statement that sets a field, then displays a report. Setting a field has a lower priority than displaying a report, so the set will normally happen after the display. 'wait' can be used between the set and display_report to force the display_report to wait until the set has finished. 'wait' should be needed rarely (if at all, see dis::7433).

Note: the various planning actions start up "background" tasks on the engine. 'wait' does not wait for these to complete (it could hang-up your gui for hours...). (some people wait 'wait' to work for these anyway, see dis::9189)

Returns: Nothing.

Parameters: - None.

Example Usage:

do(foo.set_some_value(123),

wait,

display_report("some_report", foo));

4 Summary Section

Models

Model Supply_Chain

The Fields in this Model are
name, description, sites, top_sites, sellers, top_sellers, plans.

Model Site

The Fields in this Model are
name, description, category, sub_category, rank, organization, members, member_of (Explicit_Site), role, margin_target (Date, Date), delivery_name, delivery_contact, delivery_phone, delivery_fax, delivery_address, delivery_city, delivery_state, delivery_country, delivery_postal_code, billing_name, billing_contact, billing_phone, billing_fax, billing_address, billing_city, billing_state, billing_country, billing_postal_code, tax_identification, site_plan (Plan), owner.

Model Seller

The Fields in this Model are
name, description, category, sub_category, rank, organization, members, member_of (Seller), site_groups, all_site_groups, site_group (Symbol), product_roots, products, top_products, active_strategies, product_groups, top_product_groups, forecast_horizon, first_day_of_week, week_periods, request_naming, atp_horizon, seller_plan (Plan), owner.

Model Plan

The Fields in this Model are
name, description, current, horizon, default_operation_plan_rank, site_plans, top_site_plans, seller_plans, top_seller_plans, the_problems, problems, problems (Date_Range), problems (Problem_Category), problems (Date_Range, auto_run_strategy, background_run_strategy, resource_plan (Resource), buffer_plan (Buffer), operation_plans (Operation), forecast (Product), forecast (Product_Root), run_for_now (Integer), owner.

Model Problem_Category

The Fields in this Model are
name, short_name, full_name, description, remark, super_categories, sub_categories, leaf_categories, fields.

Model Operation

The Fields in this Model are
name, description, category, sub_category, interruptible, loads, flows, consume_flows, produce_flows, all_consume_flows, all_produce_flows, all_flows, sub_operations, super_operations, process, problem_detectors, planning_buffers, planning_resources, operation_plans (Plan), unit, release_fence, release_fence_date (Plan), release_soon_fence, release_soon_fence_date (Plan), release_name_expression, next_release_number, release_number, owner.

Model Operation_Plan

The Fields in this Model are
operation, remark, rank, released, release_name, release_fence_date, release_soon_fence_date, use_alternate (Operation_Plan), use_alternate (Operation_Plan, Operation), motive, units, quantity, sid_time, expedite, dates, hint, locked_as_planned, load_plans, flow_plans, all_consuming_flow_plans, all_producing_flow_plans, sub_operation_plans, super_operation_plan, top_operation_plan, process, problems, problems (Date_Range), problems (Problem_Category), problems (Date_Range, Problem_Category), problem_categories, split (Item, Quantity, Logical), split (Quantity), Restriction), owner.

Model Resource

The Fields in this Model are
name, description, category, sub_category, location, skills, efficiency, variability, maintenance, size, load_policy, problem_detectors, resource_plan (Plan), owner.

Model Resource_Plan

The Fields in this Model are
resource, remark, efficiency_profile (Date_Range), efficiency_average (Date_Range), efficiency_average (Date), efficiency_profile_at_skill (Skill, Date_Range), efficiency_average_at_skill (Skill, Date_Range), capacity (Date_Range), efficiency, load_plans, load_plans (Date_Range), load_time (Date_Range), load_std_time (Date_Range), load_std_time (Date_Range, Logical, Logical), balance_std_time (Date_Range, Logical, Logical), move (Load_Plan, Logical, Logical, Logical), previous_gap (Load_Plan), next_gap (Load_Plan), size, size_profile (Date_Range), size_usage_profile (Date_Range), available_sized_time (Date_Range), sized_capacity (Date_Range), load_sized_time (Date_Range), load_sized_std_time (Date_Range), balance_sized_time (Date_Range, Logical, Logical), balance_sized_std_time

Summary Section

well()

(Date_Range, Logical, Logical, Logical), maintenance, problems, problems (Date_Range), problems (Problem_Category), problems (Date_Range, Problem_Category), problem_categories, problem_time (Date_Range), problem_sid_time (Date_Range), problem_sized_time (Date_Range), problem_sized_sid_time (Date_Range), resolve (Problem, Logical, Logical, Logical), resolve (Date_Range, Logical, Logical, Logical), owner.

Model Buffer

The Fields in this Model are
name, description, category, item, location, flow_policy, stocking_policy, problem_detectors, level, all_producing_operations, all_consuming_operations, unit, buffer_plan (Plan), all_supplying_operations, owner.

Model Buffer_Plan

The Fields in this Model are
buffer, remark, flow_plans, flow_plans (Date_Range), flow_plans (Date_Range, Logical), producing_flow_plans, producing_flow_plans (Date_Range), producing_flow_plans (Date_Range, Logical), consuming_flow_plans, consuming_flow_plans (Date_Range, Logical), producing_flow (Date_Range), consuming_flow (Date_Range), on_hand_profile, on_hand_profile (Date_Range), on_hand, on_hand (Date), on_hand (Date_Range), on_hand (Date_Range, Logical), on_hand (Flow_Plan), on_hand_date, on_hand_jobs (Date), lots, flow_policy, create_producing_operation_plan (Measure_Unit, Restriction), create_consuming_operation_plan (Measure_Unit, Restriction), problems, problems (Date_Range), problems (Problem_Category), problems (Date_Range, Problem_Category), problem_categories, in_flow_plans, in_flow_plans (Date_Range), in_flow_plans (Date_Range, Logical), out_flow_plans, out_flow_plans (Date_Range), out_flow_plans (Date_Range, Logical), in_flow (Date_Range), out_flow (Date_Range), owner.

Model Product_Root

The Fields in this Model are
name, description, suppliers, min_quantity, min_delivery_lead_time, delivery_area, effective_dates, customer, customers, forecast_policy, product, product (Product_Allocation), unit, owner.

Model Product

The Fields in this Model are

Summary Section

well()

product_root, name, rank, description, items, min_quantity, min_delivery_lead_time, delivery_area, customer, customers, forecast_policy, active, expiration_policy, allocation_policy, auto_allocate, always_override_members_forecasted, always_override_members_committed, availability_policy, price_policy, consume_earlier, time_pad, quantity_pad, quote_end_of_bucket, quote_multiple, quote_larger_multiple, allocate_quote_multiple, auto_allocate_from_organization, generic_products, specific_products, active_specific_products, alternate_products, primary_products, active_primary_products, groups, top, owner.

Model Forecast

The Fields in this Model are
level, name, remark, active, root, member_forecasts, active_member_forecasts, generic_forecasts, active_generic_forecasts, specific_forecasts, active_specific_forecasts, request, entries, entries_consumed, entries_members_consumed, accept_as_allocated, allocate_allocated_available, owner.

Model Site_Plan

The Fields in this Model are
site, description, organization_plan, members, role, owner.

Model Seller_Plan

The Fields in this Model are
seller, organization, members, forecasts, active_forecasts, forecast (Symbol), top_forecasts, active_top_forecasts, product_root_forecasts, active_product_root_forecasts, product_forecast (Symbol), forecast_horizon, alp_horizon, actual_requests, actual_promises, problems, problems (Date_Range), problems (Problem_Category), problems (Date_Range, Problem_Category), accept_as_allocated, owner.

Model Problem

The Fields in this Model are
description, dates, feasible, cost, last_change, category, in_category (Problem_Category), interaction, resolvable, resolve, resolve (Active_Strategy), owner.

Model Active_Strategy

The Fields in this Model are

strategy, super, remark, date_activated, reset_date_activated, run, stop, continue, running, stopped, run_time, stable_time, target_achieved, interaction, annealing_goodness, resolve_count, permanent, auto_run, background_run, active_problems, problems, problems(Date_Range), problems(Problem_Category), problems(Date_Range, Problem_Category), problem_categories, active_goals, termination, execution, problem_selection, owner.

Model Location

The Fields in this Model are
name, description, category, sub_category, super_location, sub_locations, within(Location), box, x_position, x_size_min, x_size, y_position, y_size_min, y_size, owner.

Model Item

The fields in this Model are
name, description, category, drawing_id, family, children, artificial, spec, lots_tracked, delivery, buffers, buffer_plans(Plan), unit, owner.

Model Skill

The Fields in this Model are
name, description, resources, selection, loading_operations, loading_buffers, owner.

Model Configuration

The Fields in this Model are
item, spec, interchangeable(Configuration), owner.

Model Item Group

The Fields in this Model are
name, description, top, root, sub_groups, sub_items, all_items, all_items(List(Item)), owner.

Model Operation_State

The Fields in this Model are
operation_plan, unattach, operation_plans, identifier, date, state_spec, owner.

Model Request

The Fields in this Model are

name, description, delivery_requests, delivery_policy, delivery_naming, accept_by, promise, promise_by, date_issued, date_accepted, date_queued, cancel, plan_to_satisfy, plan_as_requested, seller_plan, forecast, customer_plan, name_from_customer, delivery_name, delivery_contact, delivery_phone, delivery_fax, delivery_address, delivery_city, delivery_state, delivery_country, delivery_postal_code, last_change, issued, accepted, queued, problems, problems(Date_Range), problems(Problem_Category), problems(Date_Range, Problem_Category), owner.

Model Promise

The Fields in this Model are
request, delivery_promises, delivery_policy, acceptance, accept_by, date_offered, plan_to_satisfy, promise_as_planned, promise_as_planned(Logical), plan_as_promised, reject, last_change, problems(Date_Range), problems(Problem_Category), problems(Date_Range, Problem_Category), offered, owner.

Model Acceptance

The Fields in this Model are
promise, delivery_acceptances, accepted, accept_by, plan_to_satisfy, plan_as_accepted, accept_as_promised, accept_as_promised(Logical), last_change, problems, problems(Date_Range), problems(Problem_Category), problems(Date_Range, Problem_Category), owner.

Model Horizon

The Fields in this Model are
name, description, bucket_spec, buckets(Date_Range).

Model Horizon_Bucket_Start

The Fields in this Model are
start, owner.

Model Site_Group

The Fields in this Model are
name, description, sites, sites(List(Site)), spec, explicit_sites, owner.

Model Product_Group

The Fields in this Model are
name, description, top, root, use_sid_split, sub_groups, sub_products, all_products, all_products(List(Product)), all_products_and_specs, unit, owner.

Summary Section

wall()

Model Delivery_Request

The fields in this Model are

name, actual, forecast_entry, delivery_promise, item_requests, due, promising_policy, fulfillment_policy, max_price, seller_plan, customer_plan, name_from_customer, rank, delivery_name, delivery_contact, delivery_phone, delivery_fax, delivery_address, delivery_city, delivery_state, delivery_country, delivery_postal_code, promised_late, promised_early, promised_overpriced, cancel, plan_to_satisfy, plan_as_requested, not_planned, planned_late, planned_early, last_change, problems (Date_Range, Problem_Category), problems (Problem_Category), problems (Date_Range, Problem_Category), plan_problems, promised_date_problem, promised_price_problem, plan_problems, owner.

Model Delivery_Promise

The fields in this Model are

delivery_request, delivery_acceptance, delivery_atp, promising_policy_atp, item_promises, due, rank, fulfillment_policy, date_confirmed, list_price, sum_discount, sum_price, delivery_discount, delivery_price, promised_late, promised_early, promised_overpriced, plan_to_satisfy, plan_as_promised, promise_as_planned, promise_by_policy, not_planned, planned_late, planned_early, last_change, problems (Date_Range), problems (Problem_Category), problems (Date_Range, Problem_Category), confirmed, promised_date_problem, promised_price_problem, plan_problems, owner.

Model Delivery_Acceptance

The fields in this Model are

delivery_promise, item_acceptances, due, fulfillment_policy, plan_to_satisfy, plan_as_accepted, accept_as_promised, not_planned, planned_late, planned_early, last_change, problems (Date_Range), problems (Problem_Category), problems (Date_Range, Problem_Category), plan_problems, delivery_not_coordinated_problem, owner.

Model Strategy

The fields in this Model are

name, description, deterministic_resolvers, deterministic_problem_selection, problem_sets, changes, locks, default_change_focus, goals, termination, execution, problem_selection, max_stable_time, max_stable_resolve_count, max_run_time, max_resolve_count, max_heal, min_heal, run (Plan), do_before, do_after, do_before_resolve.

Model Problem_Set

The fields in this Model are

Summary Section

wall()

fence, min_time, last_change_after_activated, category, min_cost, must_resolve, focus, feasible_focus, horizon, owner.

Model Strategy_Change

The fields in this Model are

change_category, focus, owner.

Model Strategy_Lock

The fields in this Model are

name, spec, owner.

Model Strategy_Goal

The fields in this Model are

goal, focus_to_target, focus_beyond_target, owner.

Model Active_Problem

The fields in this Model are

problem, focus, unresolvable, interaction, must_resolve, owner.

Model Active_Goal

The fields in this Model are

strategy_goal, goal, adjusted_value, adjusted_target, focus_value, owner.

Model Explicit_Site

The fields in this Model are

site, owner.

Model Unit

The fields in this Model are

preferred_measure, discrete, quantities, convert (Measure_Unit, Measure), convert (Measure_Unit, Measure_Unit).

Model Unit_Quantity

The fields in this Model are

quantity, owner.

Model Product_Supplier

The fields in this Model are

supplier, items, owner.

Model Product_Item

The Fields in this Model are
item, owner.

Model Generic_Product
The Fields in this Model are
product, quantity_per, owner.

Model Alternatic_Product
The Fields in this Model are
product, quantity_per, rank, owner.

Model Sub_Product_Group
The Fields in this Model are
product_group, quantity_per, sid_split, root, owner.

Model Sub_Product
The Fields in this Model are
product, quantity_per, sid_split, root, owner.

Model Forecast_Entry
The Fields in this Model are
delivery_dates, date_forecasted, forecasted, cumulative_forecasted,
specifics_forecasted, members_forecasted, override_members_forecasted,
date_committed, committed, cumulative_committed, specifics_committed,
members_committed, override_members_committed, retain_from_allocated,
lock_retain_from_allocated, retain_from_accepted, planned,
cumulative_planned, specifics_planned, members_planned, date_allocated, allo-
cated, lock_allocated, cumulative_allocated, specifics_allocated,
members_allocated, date_accepted, accepted, cumulative_accepted,
members_accepted, accept_as_allocated, consumed, cumulative_consumed,
specifics_consumed, members_consumed, actual_promises,
available_to_promise, available, cumulative_available, members_available,
specifics_available, allocated_available, cumulative_allocated_available,
planned_available, cumulative_planned_available, zero_available_to_promise,
allocate_allocated_available, forecast_policy, allocation_policy, forecast_requests,
forecast_promises, owner.

Model ATP_Entry
The Fields in this Model are
available_dates, allocated, consumed, allocated_available,
cumulative_allocated_available, owner.

Model Item_Request
The Fields in this Model are
name, configuration, item, quantity, cancel, cancelled, item_promise,
mux_price, promised_short, promised_excess, promised_overpriced,
delivery_plan, plan_to_satisfy, receiving_plan, plan_as_requested, not_planned,
planned_late, planned_early, planned_short, planned_excess, last_change, prob-
lems, problems (Date_Range), problems (Problem_Category), problems
(Date_Range_Problem_Category), promised_quantity_problem,
promised_price_problem, planned_date_problem, planned_quantity_problem,
owner.

Model Load
The Fields in this Model are
name, usage_policy, resource, skill, load_sizes, start_setup, end_setup,
start_location, end_location, owner.

Model Flow
The Fields in this Model are
name, buffer, phantom, produced, usage_policy, owner.

Model Operation_Problem_Detector
The Fields in this Model are
detector, feasible, cost, owner.

Model Load_Plan
The Fields in this Model are
load, resource_plan, dates, hint, hint_on, lock_on, use_alternate,
use_alternate (Resource_Plan), owner.

Model Flow_Plan
The Fields in this Model are
flow, buffer_plan, produced, dates, quantity, quantity (Date), quantity
(Date_Range), lots, upstream_flow_plans, downstream_flow_plans, owner.

Model Item_Acceptance
The Fields in this Model are

Summary Section	wait ()
-----------------	---------

item_promise, configuration, item, quantity, delivery_plan, plan_to_satisfy, accept_as_promised, plan_as_accepted, not_planned, planned_late, planned_early, planned_short, planned_excess, last_change, problems, problems (Date_Range), problems (Problem_Category), problems (Date_Range, Problem_Category), planned_date_problem, planned_quantity_problem, owner.

Model Delivery_Available_To_Promise

The fields in this Model are
delivery_date, item_atp, offer_promise, owner.

Model Item_Promise

The Fields in this Model are
item_request, item_acceptance, item_atp, matching_forecasts, consumed_forecast, configuration, item, quantity, list_price, discount, price, promised_short, promised_excess, promised_overpriced, delivery_plan, plan_to_satisfy, promise_as_planned, plan_as_promised, receiving_plan, not_planned, planned_late, planned_early, planned_short, planned_excess, last_change, problems, problems (Date_Range), problems (Problem_Category), problems (Date_Range, Problem_Category), promised_quantity_problem, promised_price_problem, planned_date_problem, planned_quantity_problem, owner.

Model Item_Available_To_Promise

The Fields in this Model are
dates, available, product_atp, offer_promise, owner.

Model Product_Available_To_Promise

The Fields in this Model are
product, forecast, forecast_entry, dates, price, quantity, offered_quantity, offer_promise, offer_promise (Quantity), owner.

Model Buffer_Problem_Detector

The fields in this Model are
detector, feasible, cost, owner.

Model Lot_Flow

The fields in this Model are
lot, quantity, consuming_flow_plan, upstream_lot_flows, downstream_lot_flows, owner.

Model Profile_Number

Summary Section	wait ()
-----------------	---------

The Fields in this Model are
date, value, rate.

Model Profile_Percentage

The Fields in this Model are
date, value, rate.

Model Profile_Quantity

The Fields in this Model are
date, value, rate.

Model Lot

The Fields in this Model are
name, remark, quantity (Date), configuration, formed, producing_flow, consuming_flows, supplying_flow, owner.

Model Ordered_Sub_Strategy

The Fields in this Model are
sequence, strategy, owner.

Model Ranked_Sub_Strategy

The Fields in this Model are
rank, strategy, owner.

Model Sub_Item_Group

The Fields in this Model are
item_group, owner.

Model Sub_Item

The Fields in this Model are
item, owner.

Model Load_Size

The Fields in this Model are
name, load_size_usage, owner.

Model Box

The Fields in this Model are
specification, x, y, width, height, area, mid_x, mid_y, empty, unspecified.

Model Resource_Skill

Summary Section	wait()
The Fields in this Model are skill, efficiency_level (Date), efficiency_level (Date,Range), efficiency, owner.	
Model Skill_Resource The Fields in this Model are resource, efficiency_level (Date), efficiency_level (Date,Range), efficiency_profile (Date,Range), efficiency, owner.	
Model Product_Allocation The Fields in this Model are member, split, owner.	
Model Calendar_Entry The Fields in this Model are name, value, description, effective, daily_start, daily_end, day_pattern, rank, charge, owner.	
Model Calendar The Fields in this Model are name, description, entry_value, entries, entry (Date), dates (Date,Range), sub_calendars.	
Model Sub_Calendar The Fields in this Model are calendar, rank_expression, owner.	
Model Alternate_Operation The Fields in this Model are operation, quantity_per, rank, percentage, owner.	
Model Effective_Calendar_Operation The Fields in this Model are operation, quantity_per, percentage, calendar, owner.	
Model Routing_Operation The Fields in this Model are sequence_number, operation, quantity_per, owner.	
Model Consolidation The Fields in this Model are name, operation_plans, inc (Operation_Plan), dec (Operation_Plan), owner.	

Summary Section	wait()
Model Resource_Setup_Order The Fields in this Model are	
Model Resource_Blocks The Fields in this Model are	
Model Size_Dimension The Fields in this Model are name, min_size, max_size, owner.	
Model Flow_Criterion The Fields in this Model are criterion, rank, owner.	
Model Sorted_Bucket The Fields in this Model are dates, ending_on_hand, lock_ending_on_hand, producing_flow_plans, consuming_flow_plans, owner.	
Model Calendar_Plan The Fields in this Model are calendar, horizon, entry_value, entry (Date), dates (Date,Range).	
Model Worksheet The Fields in this Model are	
Model Control The Fields in this Model are	
Model Connection The Fields in this Model are	
Model User The Fields in this Model are name, full_name, responsibility, remark, organization, members, member_of (User), super_user, data_directories, current_data_directory, load_data_layouts (String), inc_new_data_layout (String,String), report_directories, reports, layouts, worksheets, styles, formats, domains, load_files, connections, active_ui, language,	

Summary Section	wait()
-----------------	--------

Model Report_Directory
The Fields in this Model are
sequence, directory, include, description, editable, owner.

Model Report
The Fields in this Model are

Model Layout
The Fields in this Model are

Model Style
The Fields in this Model are

Model Format
The Fields in this Model are
name_type, name, directory, description, editable, save, save (Pathname),
handles (Type), spec, owner.

Model Domain
The Fields in this Model are

Model Model_Type
The Fields in this Model are
name, description, owner_model, extensible, access, values (Typed_Value),
fields, extension_selectors.

Model Field
The Fields in this Model are
name, type, description, default, editable, field_type, access, after_set,
extension_selector, extensions, user_defined, value (Typed_Value), owner.

Model Extension_Selector
The Fields in this Model are
name, description, extensions, selected_fields (Symbol), owner.

Model Field_Error
The Fields in this Model are
target_model, target_field, error_value, error_input, description, source.

Model Function
The Fields in this Model are

Summary Section	wait()
-----------------	--------

Model Breakpoint
The Fields in this Model are

Extensions

Extension LINK
The Fields in this Extension are
managed, locations, top_locations, items, top_items, buffers, resources, skills,
operations, configurations, item_groups, top_item_groups, buffers_at_level (inte-
ger), operations_at_level (integer), buffer_levels, operation_levels.

Extension SUPPLIER
The Fields in this Extension are
items.

Extension CUSTOMER
The Fields in this Extension are

Extension LINK
The Fields in this Extension are
buffer_plans, resource_plans, operation_plans, operation_states, requests,
promises, acceptances, plan_to_satisfy_unanswered_requests,
plan_to_satisfy_queued_requests, plan_to_satisfy_all_requests,
plan_to_satisfy_all_promises, promise_as_planned, supply_requests,
supply_promises, supply_item_requests, supply_item_requests (Date_Range),
supply_item_requests (Item), supply_item_requests (Item, Date_Range),
item_demand (Isa(Item), Date_Range), problems, problems (Date_Range), prob-
lems (Problem_Category), problems (Problem_Category, Date_Range),
problem_categories.

Extension SUPPLIER
The Fields in this Extension are
requests, promises, acceptances.

Extension CUSTOMER
The Fields in this Extension are

Extension ONE
The Fields in this Extension are

Summary Section	wait ()
-----------------	---------

Extension MONTHS
The Fields in this Extension are

Extension WEEKS
The Fields in this Extension are
first_day_of_week, week_buckets.

Extension DAYS
The Fields in this Extension are
day_buckets.

Extension DATES
The Fields in this Extension are
bucket_starts.

Extension OPERATION_PLAN_RANK_RANGE
The Fields in this Extension are
min_rank, max_rank.

Extension OPERATION_PLAN_RANK_EXPRESSION
The Fields in this Extension are
expression.

Extension FEASIBILITY
The Fields in this Extension are
target_interaction.

Extension MINIMIZE_PROBLEM_COUNT
The Fields in this Extension are
target_problem_count.

Extension MINIMIZE_PROBLEMS
The Fields in this Extension are
target_problems.

Extension MINIMIZE_LATENESS
The Fields in this Extension are
target_lateness.

Extension WEIGHTED_LATENESS

Summary Section	wait ()
-----------------	---------

The Fields in this Extension are
target_lateness, day_late_power, quantity_late_power.

Extension MINIMIZE_SHORTNESS
The Fields in this Extension are
target_shortness.

Extension WEIGHTED_SHORTNESS
The Fields in this Extension are
target_shortness, quantity_short_power.

Extension MINIMIZE_COST
The Fields in this Extension are
target_cost.

Extension MAXIMIZE_PROFIT
The Fields in this Extension are
target_profit.

Extension MAXIMIZE_REVENUE
The Fields in this Extension are
target_revenue.

Extension NO_PROBLEMS
The Fields in this Extension are
problem_filter.

Extension TARGET_ACHIEVED
The Fields in this Extension are

Extension RESOLVE_COUNT_EXCEEDED
The Fields in this Extension are

Extension MANUAL
The Fields in this Extension are

Extension FEASIBILITY
The Fields in this Extension are
total_interaction.

Extension MINIMIZE_PROBLEM_COUNT

Summary Section	well()
-----------------	--------

The Fields in this Extension are
problem_count.

Extension MINIMIZE_PROBLEMS
The Fields in this Extension are
problems.

Extension MINIMIZE_LATENCY
The Fields in this Extension are
total_latency.

Extension WEIGHTED_LATENCY
The Fields in this Extension are
total_latency.

Extension WEIGHTED_SHORTNESS
The Fields in this Extension are
total_shortness.

Extension MINIMIZE_SHORTNESS
The Fields in this Extension are
total_shortness.

Extension MINIMIZE_COST
The Fields in this Extension are
total_cost.

Extension MAXIMIZE_PROFIT
The Fields in this Extension are
total_profit.

Extension MAXIMIZE_REVENUE
The Fields in this Extension are
total_revenue.

Extension NO_PROBLEMS
The Fields in this Extension are

Extension TARGET_ACHIEVED
The Fields in this Extension are

Summary Section	well()
-----------------	--------

Extension RESOLVE_COUNT_EXCEEDED
The Fields in this Extension are

Extension MANUAL
The Fields in this Extension are
done.

Extension INDIVIDUAL
The Fields in this Extension are
product, atp_entries.

Extension GROUP
The Fields in this Extension are
product_group, sub_forecasts, active_sub_forecasts, active_leaf_product_forecasts,
use_std_split.

Extension MOVE_IN
The Fields in this Extension are

Extension MOVE_OUT
The Fields in this Extension are

Extension SPLIT
The Fields in this Extension are

Extension USE_MORE
The Fields in this Extension are

Extension USE_LESS
The Fields in this Extension are

Extension USE_ALTERNATE_OPERATION
The Fields in this Extension are
min_alt, max_alt.

Extension USE_ALTERNATE_RESOURCE
The Fields in this Extension are

Extension USE_EFFECTIVE_ALTERNATE
The Fields in this Extension are

Summary Section	wait()
-----------------	--------

Extension **INCREASE_CAPACITY**
The Fields in this Extension are

Extension **DECREASE_CAPACITY**
The Fields in this Extension are

Extension **RESIZE_MORE**
The Fields in this Extension are

Extension **RESIZE_LESS**
The Fields in this Extension are

Extension **UPSTREAM**
The Fields in this Extension are

Extension **DOWNSTREAM**
The Fields in this Extension are

Extension **REQUEST_NOT_PLANNED**
The Fields in this Extension are
item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance, delivery_plan.

Extension **REQUEST_NOT_PLANNED**
The Fields in this Extension are
item_request,filter, problem_filter.

Extension **REQUEST_PLANNED_LATE**
The Fields in this Extension are
item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance, delivery_plan.

Extension **REQUEST_PLANNED_LATE**
The Fields in this Extension are
min_latency, item_request,filter, problem_filter.

Extension **REQUEST_PLANNED_EARLY**
The Fields in this Extension are

Summary Section	wait()
-----------------	--------

item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance, delivery_plan.

Extension **REQUEST_PLANNED_EARLY**
The Fields in this Extension are
min_earliness, item_request,filter, problem_filter.

Extension **REQUEST_PLANNED_SHORT**
The Fields in this Extension are
item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance, delivery_plan.

Extension **REQUEST_PLANNED_SHORT**
The Fields in this Extension are
min_shortness, item_request,filter, problem_filter.

Extension **REQUEST_PLANNED_EXCESS**
The Fields in this Extension are
item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance, delivery_plan.

Extension **REQUEST_PLANNED_EXCESS**
The Fields in this Extension are
min_excess, item_request,filter, problem_filter.

Extension **PROMISE_NOT_PLANNED**
The Fields in this Extension are
item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance, delivery_plan.

Extension **PROMISE_NOT_PLANNED**
The Fields in this Extension are
item_promise,filter, problem_filter.

Extension **PROMISE_PLANNED_LATE**
The Fields in this Extension are

Summary Section	wait()
-----------------	--------

item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance, delivery_plan.

Extension PROMISE_PLANNED_LATE
The Fields in this Extension are
min_lateness, item_promise_filter, problem_filter.

Extension PROMISE_PLANNED_EARLY
The Fields in this Extension are
item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance, delivery_plan.

Extension PROMISE_PLANNED_EARLY
The Fields in this Extension are
min_earliness, item_promise_filter, problem_filter.

Extension PROMISE_PLANNED_SHORT
The Fields in this Extension are
item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance, delivery_plan.

Extension PROMISE_PLANNED_SHORT
The Fields in this Extension are
min_shortness, item_promise_filter, problem_filter.

Extension PROMISE_PLANNED_EXCESS
The Fields in this Extension are
item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance, delivery_plan.

Extension PROMISE_PLANNED_EXCESS
The Fields in this Extension are
min_excess, item_promise_filter, problem_filter.

Extension ACCEPTANCE_NOT_PLANNED
The Fields in this Extension are

Summary Section	wait()
-----------------	--------

item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance, delivery_plan.

Extension ACCEPTANCE_NOT_PLANNED
The Fields in this Extension are
item_acceptance_filter, problem_filter.

Extension ACCEPTANCE_PLANNED_LATE
The Fields in this Extension are
item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance, delivery_plan.

Extension ACCEPTANCE_PLANNED_LATE
The Fields in this Extension are
min_lateness, item_acceptance_filter, problem_filter.

Extension ACCEPTANCE_PLANNED_EARLY
The Fields in this Extension are
item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance, delivery_plan.

Extension ACCEPTANCE_PLANNED_EARLY
The Fields in this Extension are
min_earliness, item_acceptance_filter, problem_filter.

Extension ACCEPTANCE_PLANNED_SHORT
The Fields in this Extension are
item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance, delivery_plan.

Extension ACCEPTANCE_PLANNED_SHORT
The Fields in this Extension are
min_shortness, item_acceptance_filter, problem_filter.

Extension ACCEPTANCE_PLANNED_EXCESS
The Fields in this Extension are

Summary Section	wait()
-----------------	--------

item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance, delivery_plan.

Extension ACCEPTANCE_PLANNED_EXCESS

The Fields in this Extension are
min_excess, item_acceptance_filter, problem_filter.

Extension SEQUENCE_RUN_ONCE

The Fields in this Extension are
sub_strategies.

Extension SEQUENCE_RUN_MULTIPLE

The Fields in this Extension are
sub_strategies.

Extension SEQUENCE_RUN_ONCE

The Fields in this Extension are
sub_strategies, current_sub.

Extension SEQUENCE_RUN_MULTIPLE

The Fields in this Extension are
sub_strategies, current_sub.

Extension BEFORE_AND_AFTER

The Fields in this Extension are
before_run, before_search, after_resolve, after_search, after_success, after_run.

Extension BEFORE_AND_AFTER

The Fields in this Extension are
before_run, before_search, after_resolve, after_search, after_success, after_run.

Extension SEQUENTIAL_ALTERNATES

The Fields in this Extension are
propagation, evaluation, cleanup, min_all, max_all.

Extension SEQUENTIAL_ALTERNATES

The Fields in this Extension are
propagation, evaluation, cleanup, min_all, max_all, current_all.

Extension SEQUENTIAL_ALTERNATES_KEEP_BEST

Summary Section	wait()
-----------------	--------

The Fields in this Extension are
propagation, evaluation, min_all, max_all.

Extension SEQUENTIAL_ALTERNATES_KEEP_BEST

The Fields in this Extension are
propagation, evaluation, min_all, max_all.

Extension PROPORTIONAL_RESOLVES

The Fields in this Extension are
ranked_sub_strategies.

Extension PROPORTIONAL_RESOLVES

The Fields in this Extension are
sub_strategies.

Extension ORDERED_RESOLVES

The Fields in this Extension are
sub_strategies.

Extension ORDERED_RESOLVES

The Fields in this Extension are
sub_strategies.

Extension PRODUCE

The Fields in this Extension are
buffer_plan, configuration.

Extension CONSUME

The Fields in this Extension are
buffer_plan, configuration.

Extension DELIVER

The Fields in this Extension are
motive_request, motive_promise, planned_for_promise, item, configuration.

Extension RECEIVE

The Fields in this Extension are
item.

Extension PLANNED_BEFORE_CURRENT

The Fields in this Extension are

Summary Section	wait()
-----------------	--------

operation_plan.

Extension **PLANNED_BEFORE_CURRENT**
The Fields in this Extension are
operation_plan_filter, problem_filter.

Extension **UNRELEASED**
The Fields in this Extension are
operation_plan.

Extension **UNRELEASED**
The Fields in this Extension are
operation_plan_filter, problem_filter.

Extension **NEEDS_RELEASE**
The Fields in this Extension are
operation_plan.

Extension **NEEDS_RELEASE**
The Fields in this Extension are
operation_plan_filter, problem_filter.

Extension **INCONSISTENT_OPPLAN**
The Fields in this Extension are
operation_plan.

Extension **INCONSISTENT_OPPLAN**
The Fields in this Extension are
operation_plan_filter, problem_filter.

Extension **OPERATION**
The Fields in this Extension are
operation_plan_filter, problem_filter.

Extension **REQUEST_PROMISED_LATE**
The Fields in this Extension are
delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance.

Extension **REQUEST_PROMISED_LATE**
The Fields in this Extension are

Summary Section	wait()
-----------------	--------

min_lateness, delivery_promise_filter, problem_filter.

Extension **REQUEST_PROMISED_EARLY**
The Fields in this Extension are
delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance.

Extension **REQUEST_PROMISED_EARLY**
The Fields in this Extension are
min_earliness, delivery_promise_filter, problem_filter.

Extension **REQUEST_PROMISED_SHORT**
The Fields in this Extension are
item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance.

Extension **REQUEST_PROMISED_SHORT**
The Fields in this Extension are
min_shortness, item_promise_filter, problem_filter.

Extension **REQUEST_PROMISED_EXCESS**
The Fields in this Extension are
item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance.

Extension **REQUEST_PROMISED_EXCESS**
The Fields in this Extension are
min_excess, item_promise_filter, problem_filter.

Extension **ITEM_PROMISE_OVERPRICED**
The Fields in this Extension are
item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance.

Extension **DELIVERY_PROMISE_OVERPRICED**
The Fields in this Extension are
delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance.

Extension **PROMISE_NOT_OFFERED**
The Fields in this Extension are

request, promise, acceptance.

Extension PROMISE_NOT_OFFERED

The Fields in this Extension are
request_filter, problem_filter.

Extension PROMISE_NOT_ACCEPTED

The Fields in this Extension are
request, promise, acceptance.

Extension PROMISE_NOT_ACCEPTED

The Fields in this Extension are
request_filter, problem_filter.

Extension ACCEPTANCE_INCONSISTENT

The Fields in this Extension are
request, promise, acceptance.

Extension ACCEPTANCE_INCONSISTENT

The Fields in this Extension are
request_filter, problem_filter.

Extension REQUEST_QUEUED

The Fields in this Extension are
request, promise, acceptance.

Extension REQUEST_QUEUED

The Fields in this Extension are
request_filter, problem_filter.

Extension DELIVERY_REQUEST_NOT_COORDINATED

The Fields in this Extension are
delivery_request, delivery_promise, delivery_acceptance, request, promise,
acceptance.

Extension DELIVERY_PROMISE_NOT_COORDINATED

The Fields in this Extension are
delivery_request, delivery_promise, delivery_acceptance, request, promise,
acceptance.

Extension DELIVERY_ACCEPTANCE_NOT_COORDINATED

The Fields in this Extension are
delivery_request, delivery_promise, delivery_acceptance, request, promise,
acceptance.

Extension REQUEST

The Fields in this Extension are
request_filter, problem_filter.

Extension REQUEST_PLAN

The Fields in this Extension are
item_request_filter, problem_filter.

Extension PROMISE

The Fields in this Extension are
promise_filter, problem_filter.

Extension PROMISE_PLAN

The Fields in this Extension are
item_promise_filter, problem_filter.

Extension REQUEST_PROMISE

The Fields in this Extension are
delivery_promise_filter, problem_filter.

Extension DELIVERY_REQUEST_NOT_COORDINATED

The Fields in this Extension are
problem_filter.

Extension DELIVERY_PROMISE_NOT_COORDINATED

The Fields in this Extension are
problem_filter.

Extension DELIVERY_ACCEPTANCE_NOT_COORDINATED

The Fields in this Extension are
problem_filter.

Extension NEGATIVE_ATP

The Fields in this Extension are
forecast_entry.

Extension NEGATIVE_PLANNED_ATP

Summary Section	Wall ()
-----------------	----------

The Fields in this Extension are
forecast_entry.

Extension OVER_COMMITTED
The Fields in this Extension are
forecast_entry.

Extension OVER_CONSUMED
The Fields in this Extension are
forecast_entry.

Extension UNALLOCATED_FORECAST
The Fields in this Extension are
forecast_entry.

Extension PROPORTIONAL_INTERACTION
The Fields in this Extension are

Extension EARLIEST_PROBLEM_START
The Fields in this Extension are

Extension SORT_BY_EXPRESSION
The Fields in this Extension are

Extension PROPORTIONAL_INTERACTION
The Fields in this Extension are

Extension EARLIEST_PROBLEM_START
The Fields in this Extension are

Extension SORT_BY_EXPRESSION
The Fields in this Extension are
sorting_criteria.

Extension SUPPLY_PLANNED_LATE
The Fields in this Extension are
receiving_plan, item_request, item_promise, item_acceptance, delivery_request,
delivery_promise, delivery_acceptance, request, promise, acceptance,
delivery_plan.

Extension SUPPLY_PLANNED_LATE

Summary Section	Wall ()
-----------------	----------

The Fields in this Extension are
min_latency, item_request_filter, problem_filter.

Extension SUPPLY_PLANNED_EARLY
The Fields in this Extension are
receiving_plan, item_request, item_promise, item_acceptance, delivery_request,
delivery_promise, delivery_acceptance, request, promise, acceptance,
delivery_plan.

Extension SUPPLY_PLANNED_EARLY
The Fields in this Extension are
min_earliness, item_request_filter, problem_filter.

Extension SUPPLY_PLANNED_SHORT
The Fields in this Extension are
receiving_plan, item_request, item_promise, item_acceptance, delivery_request,
delivery_promise, delivery_acceptance, request, promise, acceptance,
delivery_plan.

Extension SUPPLY_PLANNED_SHORT
The Fields in this Extension are
min_shortness, item_request_filter, problem_filter.

Extension SUPPLY_PLANNED_EXCESS
The Fields in this Extension are
receiving_plan, item_request, item_promise, item_acceptance, delivery_request,
delivery_promise, delivery_acceptance, request, promise, acceptance,
delivery_plan.

Extension SUPPLY_PLANNED_EXCESS
The Fields in this Extension are
min_excess, item_request_filter, problem_filter.

Extension SUPPLY_PROMISED_LATE
The Fields in this Extension are
receiving_plan, item_request, item_promise, item_acceptance, delivery_request,
delivery_promise, delivery_acceptance, request, promise, acceptance.

Extension SUPPLY_PROMISED_LATE
The Fields in this Extension are
min_latency, delivery_promise_filter, problem_filter.

Summary Section	wait ()
-----------------	---------

Extension SUPPLY_PROMISED_EARLY

The Fields in this Extension are
 receiving_plan, item_request, item_promise, item_acceptance, delivery_request,
 delivery_promise, delivery_acceptance, request, promise, acceptance.

Extension SUPPLY_PROMISED_EARLY

The Fields in this Extension are
 min_curliness, delivery_promise_filter, problem_filter.

Extension SUPPLY_PROMISED_SHORT

The Fields in this Extension are
 receiving_plan, item_request, item_promise, item_acceptance, delivery_request,
 delivery_promise, delivery_acceptance, request, promise, acceptance.

Extension SUPPLY_PROMISED_SHORT

The Fields in this Extension are
 min_shortness, item_promise_filter, problem_filter.

Extension SUPPLY_PROMISED_EXCESS

The Fields in this Extension are
 receiving_plan, item_request, item_promise, item_acceptance, delivery_request,
 delivery_promise, delivery_acceptance, request, promise, acceptance.

Extension SUPPLY_PROMISED_EXCESS

The Fields in this Extension are
 min_excess, item_promise_filter, problem_filter.

Extension SUPPLY

The Fields in this Extension are
 request_filter, problem_filter.

Extension SUPPLY_PLAN

The Fields in this Extension are
 item_request_filter, problem_filter.

Extension SUPPLY_PROMISE

The Fields in this Extension are
 delivery_promise_filter, problem_filter.

Extension FCFS

Summary Section	wait ()
-----------------	---------

The Fields in this Extension are

Extension PER_ALLOCATED

The Fields in this Extension are
 retain,

Extension PER_COMMITTED

The Fields in this Extension are
 retain,

Extension MEMBER_RANK

The Fields in this Extension are
 retain, pool_allocation, reallocate (Plan).

Extension MEMBER_RANK

The Fields in this Extension are
 pooled_allocated, pooled_accepted, pooled_allocated_available, pooled_available.

Extension FIXED_SPLIT

The Fields in this Extension are
 retain, allocations.

Extension SLIDING

The Fields in this Extension are

Extension HORIZON

The Fields in this Extension are
 availability_horizon, buckets (Date_Range).

Extension BUCKETED_ALL

The Fields in this Extension are

Extension BUCKETED_ASAP

The Fields in this Extension are

Extension FIXED

The Fields in this Extension are

Extension FIXED

The Fields in this Extension are

Summary Section	wait()
-----------------	--------

Extension **AT_END**
The Fields in this Extension are

Extension **SIMPLE_FIXED_QUANTITY**
The Fields in this Extension are
order_quantity, representative_configuration, representative_quantity_per,
rough_fence.

Extension **SIMPLE_FIXED_QUANTITY**
The Fields in this Extension are

Extension **SINGLE_REQUEST**
The Fields in this Extension are
representative_configuration, representative_quantity_per.

Extension **SINGLE_REQUEST**
The Fields in this Extension are

Extension **SIMPLE_FIXED_TIME**
The Fields in this Extension are
interval, representative_configuration, representative_quantity_per, rough_fence.

Extension **SIMPLE_FIXED_TIME**
The Fields in this Extension are

Extension **WEEKLY**
The Fields in this Extension are
day_of_week, representative_configuration, representative_quantity_per,
rough_fence.

Extension **WEEKLY**
The Fields in this Extension are

Extension **DUAL_REQUEST**
The Fields in this Extension are
representative_configuration, representative_quantity_per, rough_fence.

Extension **DUAL_REQUEST**
The Fields in this Extension are

Extension **ON_TIME**

Summary Section	wait()
-----------------	--------

The Fields in this Extension are

Extension **ON_TIME**
The Fields in this Extension are

Extension **ON_TIME**
The Fields in this Extension are

Extension **FULL_QUANTITIES_OF_ALL_ITEMS**
The Fields in this Extension are

Extension **FULL_QUANTITIES_OF_ALL_ITEMS**
The Fields in this Extension are

Extension **FULL_QUANTITIES_OF_ALL_ITEMS**
The Fields in this Extension are

Extension **UNRESTRICTED**
The Fields in this Extension are

Extension **UNRESTRICTED**
The Fields in this Extension are

Extension **UNRESTRICTED**
The Fields in this Extension are

Extension **NUMBERED**
The Fields in this Extension are

Extension **NONE**
The Fields in this Extension are

Extension **NONE**
The Fields in this Extension are

Extension **FIXED**
The Fields in this Extension are
price.

Extension **ON_TIME**
The Fields in this Extension are

Extension ALL

The Fields in this Extension are

Extension ALL_ON_TIME

The Fields in this Extension are

Extension ASAP

The Fields in this Extension are

Extension ASAP_MONTHLY

The Fields in this Extension are

Extension BUCKETED_ALLOCATION

The Fields in this Extension are

Extension BUCKETED_ALL_MIN_PRICE

The Fields in this Extension are

Extension BUCKETED_MIN_PRICE_ASAP

The Fields in this Extension are

Extension SHIP_IN_RATIO

The Fields in this Extension are

Extension UNSPECIFIED

The Fields in this Extension are

Extension NUMBER

The Fields in this Extension are
default_number.

Extension QUANTITY

The Fields in this Extension are
default_quantity.

Extension NUMBER_QUANTITY

The Fields in this Extension are
default_number, default_quantity.

Extension SYMBOL

The Fields in this Extension are
default_value.

Extension TIME

The Fields in this Extension are
default_time.

Extension ALTERNATES_PRIMARY

The Fields in this Extension are
splittable, alternates.

Extension ALTERNATES_PRIMARY

The Fields in this Extension are
move_to_alternate (Operation_Plan, Operation, Quantity), move_to_alternate
(Operation_Plan, Operation), move_to_alternate (Operation_Plan),
move_to_alternate.

Extension ALTERNATES_PROPORTIONAL

The Fields in this Extension are
splittable, alternates.

Extension ALTERNATES_PROPORTIONAL

The Fields in this Extension are
move_to_alternate (Operation_Plan, Operation, Quantity), move_to_alternate
(Operation_Plan, Operation), move_to_alternate (Operation_Plan),
move_to_alternate.

Extension EFFECTIVE_CALENDAR

The Fields in this Extension are
splittable, effective_alternates.

Extension EFFECTIVE_CALENDAR

The Fields in this Extension are
move_to_alternate (Operation_Plan, Operation, Quantity), move_to_alternate
(Operation_Plan, Operation), move_to_alternate (Operation_Plan),
move_to_alternate.

Extension CONSUME_FIXED

The Fields in this Extension are
fixed_quantity.

Extension CONSUME_PER
The Fields in this Extension are
quantity_per.

Extension PRODUCE_FIXED
The Fields in this Extension are
fixed_quantity.

Extension PRODUCE_PER
The Fields in this Extension are
quantity_per.

Extension PRODUCE_YIELD
The Fields in this Extension are
quantity_per, yield, yield_near, near_time.

Extension PRODUCE_YIELD_CALENDAR
The Fields in this Extension are
quantity_per, yield, yield_near, near_time, calendar.

Extension EARLIEST
The Fields in this Extension are
release_name, operation, top_operation.

Extension FIXED
The Fields in this Extension are
fixed_quantity.

Extension LINEAR
The Fields in this Extension are
linear_quantity.

Extension RESOURCE
The Fields in this Extension are

Extension ONE
The Fields in this Extension are

Extension UNIDENTIFIED_OP_STATE
The Fields in this Extension are
operation_state.

Extension UNIDENTIFIED_OP_STATE
The Fields in this Extension are
operation_plan_filter, problem_filter.

Extension DELAY_ONLY_FIXED
The Fields in this Extension are
time.

Extension DELAY_ONLY_BASIC
The Fields in this Extension are

Extension DELAY_ONLY_BASIC
The Fields in this Extension are
time, time_per.

Extension DELAY_ONLY_BASIC
The Fields in this Extension are

Extension BASIC_CALENDARS
The Fields in this Extension are
time_calendar, time_per_calendar.

Extension BASIC_CALENDARS
The Fields in this Extension are

Extension FIXED_TIME
The Fields in this Extension are
time.

Extension FIXED_TIME
The Fields in this Extension are

Extension TIME_MULTIPLE
The Fields in this Extension are
base_time, base_units.

Extension TIME_MULTIPLE
The Fields in this Extension are

Extension BASIC

Summary Section	wait ()
-----------------	----------

The Fields in this Extension are
time, time_per.

Extension BASIC
The Fields in this Extension are

Extension BASIC_DELAYED
The Fields in this Extension are
time, time_per, pre_load_delay, post_load_delay.

Extension BASIC_DELAYED
The Fields in this Extension are

Extension REQUEST_FIXED
The Fields in this Extension are
time, order_lead_time, item, supplier, item_name, seller.

Extension REQUEST_FIXED
The Fields in this Extension are
item_request.

Extension REQUEST_FIXED_WITH_ANALYSIS
The Fields in this Extension are
time, order_lead_time, item, supplier, item_name, seller.

Extension REQUEST_FIXED_WITH_ANALYSIS
The Fields in this Extension are
item_request.

Extension ROUTING
The Fields in this Extension are
routing_operations.

Extension ROUTING
The Fields in this Extension are

Extension STARTED
The Fields in this Extension are
item, quantity.

Extension COMPLETED

Summary Section	wait ()
-----------------	----------

The Fields in this Extension are
item, quantity.

Extension IN_FRONT
The Fields in this Extension are
quantity.

Extension SIMPLE_CONSOLIDATION
The Fields in this Extension are
consolidation_fence.

Extension SIMPLE_CONSOLIDATION
The Fields in this Extension are
consolidations, unconsolidated_operation_plans.

Extension UNCONSOLIDATED
The Fields in this Extension are
resource_plan, operation_plans.

Extension UNCONSOLIDATED
The Fields in this Extension are
resource_plan_filter.

Extension UNCOORDINATED
The Fields in this Extension are
resource_plan, operation_plans, consolidation.

Extension UNCOORDINATED
The Fields in this Extension are
resource_plan_filter.

Extension CONSOLIDATION_OVERSIZE
The Fields in this Extension are
resource_plan, operation_plans, consolidation, oversize_dimensions, oversize (Symbol).

Extension CONSOLIDATION_OVERSIZE
The Fields in this Extension are
resource_plan_filter,

Extension CONSOLIDATION_UNDERSIZE

The fields in this Extension are
`resource_plan`, `operation_plans`, `consolidation`, `undersize_dimensions`, `undersize` (Symbol).

Extension **CONSOLIDATION_UNDERSIZE**
The Fields in this Extension are
`resource_plan_filter`,

Extension **FIXED**
The Fields in this Extension are
`fixed_efficiency`.

Extension **CALENDAR**
The Fields in this Extension are
`efficiency_calendar`.

Extension **INFINITE_USE**
The Fields in this Extension are

Extension **INFINITE_USE**
The Fields in this Extension are

Extension **EXCLUSIVE_USE**
The Fields in this Extension are

Extension **EXCLUSIVE_USE**
The Fields in this Extension are

Extension **SHARED_USE**
The Fields in this Extension are
`buckets`, `bucket_fence`, `min_load`, `min_load_fence`, `max_bucket_load`,
`upstream_bucket_pad`, `downstream_bucket_pad`,

Extension **SHARED_USE**
The Fields in this Extension are

Extension **ZERO**
The Fields in this Extension are

Extension **OVERLOAD**
The Fields in this Extension are

`resource_plan`, `overload_time`, `overload_std_time`.

Extension **OVERLOAD**
The Fields in this Extension are
`resource_plan_filter`, `problem_filter`, `min_overload_time`, `min_overload_std_time`.

Extension **OVERSIZE**
The Fields in this Extension are
`resource_plan`, `oversize`.

Extension **OVERSIZE**
The Fields in this Extension are
`resource_plan_filter`, `problem_filter`, `min_oversize`.

Extension **BUCKET_OVERSIZE**
The Fields in this Extension are
`resource_plan`, `bucket_oversize`.

Extension **BUCKET_OVERSIZE**
The Fields in this Extension are
`resource_plan_filter`, `min_bucket_oversize`.

Extension **UNDERLOAD**
The Fields in this Extension are
`resource_plan`, `underload`.

Extension **UNDERLOAD**
The Fields in this Extension are
`resource_plan_filter`, `min_underload`.

Extension **RESOURCE**
The Fields in this Extension are
`resource_plan_filter`, `problem_filter`.

Extension **FIXED**
The Fields in this Extension are
`fixed_efficiency`.

Extension **CALENDAR**
The Fields in this Extension are
`efficiency_calendar`.

Summary Section	wait ()
-----------------	---------

Extension PRIMARY
The Fields in this Extension are primary.

Extension PREFER_PRIMARY
The Fields in this Extension are primary.

Extension EVEN
The Fields in this Extension are

Extension MAX_EFFICIENCY
The Fields in this Extension are

Extension UNLIMITED
The Fields in this Extension are

Extension FIXED_COUNT
The Fields in this Extension are max_operation_count.

Extension FIXED_QUANTITY
The Fields in this Extension are max_size.

Extension CALENDAR_COUNT
The Fields in this Extension are size_calendar.

Extension MULTI_DIMENSION
The Fields in this Extension are dimensions.

Extension FIXED
The Fields in this Extension are fixed_efficiency.

Extension CALENDAR
The Fields in this Extension are efficiency_calendar.

Summary Section	wait ()
-----------------	---------

Extension ZERO
The Fields in this Extension are

Extension FIXED
The Fields in this Extension are upstream_pad, downstream_pad.

Extension NEGATIVE_ON_HAND
The Fields in this Extension are buffer_plan, low_on_hand, deficit.

Extension NEGATIVE_ON_HAND
The Fields in this Extension are buffer_plan_filter, problem_filter, min_deficit.

Extension OVER_FLOW_LIMIT
The Fields in this Extension are buffer_plan, low_on_hand, deficit.

Extension OVER_FLOW_LIMIT
The Fields in this Extension are buffer_plan_filter, problem_filter, min_deficit.

Extension NEGATIVE_ON_HAND_AT_END
The Fields in this Extension are buffer_plan, low_on_hand, deficit.

Extension NEGATIVE_ON_HAND_AT_END
The Fields in this Extension are buffer_plan_filter, problem_filter, min_deficit.

Extension LOT_OVER_CONSUMED
The Fields in this Extension are buffer_plan, shortage_quantity, shortage_time.

Extension LOT_OVER_CONSUMED
The Fields in this Extension are buffer_plan_filter, problem_filter.

Extension LOT_NOT_CONSUMED

Summary Section	wait()
-----------------	--------

The Fields in this Extension are
buffer_plan.

Extension LOT_NOT_CONSUMED
The Fields in this Extension are
buffer_plan_filter, problem_filter.

Extension LOT_NOT_PRODUCED
The Fields in this Extension are
buffer_plan.

Extension LOT_NOT_PRODUCED
The Fields in this Extension are
buffer_plan_filter, problem_filter.

Extension LOT_OVER_PRODUCED
The Fields in this Extension are
buffer_plan, excess_quantity, excess_time.

Extension LOT_OVER_PRODUCED
The Fields in this Extension are
buffer_plan_filter, problem_filter, min_excess_time.

Extension LOW_ON_HAND
The Fields in this Extension are
buffer_plan, low_on_hand, deficit.

Extension LOW_ON_HAND
The Fields in this Extension are
buffer_plan_filter, problem_filter, min_deficit.

Extension EXCESS_ON_HAND
The Fields in this Extension are
buffer_plan, excess.

Extension EXCESS_ON_HAND
The Fields in this Extension are
buffer_plan_filter, problem_filter, min_excess.

Extension EXCESS_ON_HAND_AT_END
The Fields in this Extension are

Summary Section	wait()
-----------------	--------

buffer_plan, excess.

Extension EXCESS_ON_HAND_AT_END
The Fields in this Extension are
buffer_plan_filter, problem_filter, min_excess.

Extension BUFFER
The Fields in this Extension are
buffer_plan_filter, problem_filter.

Extension BUCKETED_NESTED_SORT
The Fields in this Extension are
producing_operation, horizon, fixed_ending_on_hand, per_next_ending_on_hand, default_product_root, do_not_move_out_forecasts, split_multiple, criteria.

Extension BUCKETED_NESTED_SORT
The Fields in this Extension are
sorted_buckets, allocate_excess.

Extension PRODUCING_FLOW_CALENDAR
The Fields in this Extension are
calendar, quantity_range, multiple_quantity, fence, after_fence_quantity_range, after_fence_multiple_quantity, default_min_on_hand, min_on_hand_calendar, default_excess_on_hand, excess_on_hand_calendar, default_min_time, min_time_calendar, end_item, rank_delivery_operation_plan_higher, producing_operation.

Extension PRODUCING_FLOW_CALENDAR
The fields in this Extension are
max_target_profile, max_target_profile(Date,Range), max_target(Date), safety_target_profile(Date,Range), safety_target(Date), safety_target(Date,Range), safety_target(Date,Range,Logical), cycle_on_hand_profile, cycle_on_hand_profile(Date,Range), cycle_on_hand(Date), cycle_on_hand(Date,Range), cycle_on_hand(Date,Range,Logical), excess_on_hand_profile, excess_on_hand_profile(Date,Range), excess_on_hand(Date), excess_on_hand(Date,Range), excess_on_hand(Date,Range,Logical), low_on_hand_profile, low_on_hand_profile(Date,Range), low_on_hand(Date), low_on_hand(Date,Range), low_on_hand(Date,Range,Logical).

Extension PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK

The Fields in this Extension are

calendar, quantity_range, multiple_quantity, fence, after_fence_quantity_range, after_fence_multiple_quantity, default_min_on_hand, min_on_hand_calendar, default_excess_on_hand, excess_on_hand_calendar, default_min_time, min_time_calendar, end_item, rank, delivery_operation_plan_higher, producing_operation, flow_plan_selection_start, flow_plan_selection_end, flow_plan_selection_interval, flow_plan_filter, flow_plan_rank, continue_flow_plan_selection, flow_plan_resize_quantity_range, flow_plan_move_restriction, flow_plan_split_restriction.

Extension PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK

The Fields in this Extension are

max_target_profile, max_target_profile (Date_Range), max_target (Date), max_target (Date_Range), max_target (Date_Range_Logical), safety_target_profile, safety_target_profile (Date_Range), safety_target (Date), safety_target (Date_Range), safety_target (Date_Range_Logical), cycle_on_hand_profile, cycle_on_hand_profile (Date_Range), cycle_on_hand (Date), cycle_on_hand (Date_Range), cycle_on_hand (Date_Range_Logical), excess_on_hand_profile, excess_on_hand_profile (Date_Range), excess_on_hand (Date), excess_on_hand (Date_Range), excess_on_hand (Date_Range_Logical), low_on_hand_profile, low_on_hand_profile (Date_Range), low_on_hand (Date), low_on_hand (Date_Range), low_on_hand (Date_Range_Logical).

Extension SUPPLY_CALENDAR

The Fields in this Extension are

calendar.

Extension ON_HAND_CALENDAR

The Fields in this Extension are

calendar.

Extension ON_HAND_CALENDAR

The Fields in this Extension are

capacity (Date_Range).

Extension ON_HAND_CALENDAR_FILTER_AND_RANK

The Fields in this Extension are

calendar, flow_plan_selection_start, flow_plan_selection_end, flow_plan_selection_interval, flow_plan_filter, flow_plan_rank, continue_flow_plan_selection, flow_plan_resize_quantity_range, flow_plan_move_restriction, flow_plan_split_restriction.

Extension ON_HAND_CALENDAR_FILTER_AND_RANK

The Fields in this Extension are

capacity (Date_Range).

Extension FLOW_LIMIT_CALENDAR

The Fields in this Extension are

calendar.

Extension FLOW_LIMIT_CALENDAR

The Fields in this Extension are

capacity (Date_Range).

Extension FLOW_LIMIT_CALENDAR_FILTER_AND_RANK

The Fields in this Extension are

calendar, flow_plan_selection_start, flow_plan_selection_end, flow_plan_selection_interval, flow_plan_filter, flow_plan_rank, continue_flow_plan_selection, flow_plan_resize_quantity_range, flow_plan_move_restriction, flow_plan_split_restriction.

Extension FLOW_LIMIT_CALENDAR_FILTER_AND_RANK

The Fields in this Extension are

capacity (Date_Range).

Extension CUSTOMER_RANK

The Fields in this Extension are

produce_separately, default_rank.

Extension SELLER_RANK

The Fields in this Extension are

produce_separately, default_rank.

Extension REQUEST_RANK

The Fields in this Extension are

produce_separately, default_rank.

Extension ACTUAL_OR_FORECAST

The Fields in this Extension are

produce_separately, default_rank.

Extension REQUEST_ISSUED

The Fields in this Extension are

Summary Section	wait()
-----------------	--------

produce_separately, default_rank, default_issued.

Extension PROMISE_DUE

The Fields in this Extension are
produce_separately, default_rank, default_due.

Extension REQUEST_DUE

The Fields in this Extension are
produce_separately, default_rank, default_due.

Extension DUE_DATE

The Fields in this Extension are
produce_separately, default_rank, default_due.

Extension PROMISED

The Fields in this Extension are
produce_separately, default_rank.

Extension ENTRY_DATE

The Fields in this Extension are
produce_separately, default_rank.

Extension INFINITE

The Fields in this Extension are

Extension SUPPLIER

The Fields in this Extension are
fence.

Extension BASIC

The Fields in this Extension are
min_time, min_on_hand, excess_on_hand, producing_operation,
supplying_operation.

Extension BASIC

The Fields in this Extension are
max_target_profile, max_target_profile(Date_Range), max_target(Date),
max_target(Date_Range), max_target(Date_Range, Logical), safety_target_profile,
safety_target_profile(Date_Range), safety_target(Date), safety_target
(Date_Range), safety_target(Date_Range, Logical), cycle_on_hand_profile,
cycle_on_hand_profile(Date_Range), cycle_on_hand(Date), cycle_on_hand

Summary Section	wait()
-----------------	--------

(Date_Range), cycle_on_hand(Date_Range, Logical), excess_on_hand_profile,
excess_on_hand_profile(Date_Range), excess_on_hand(Date), excess_on_hand
(Date_Range), excess_on_hand(Date_Range, Logical), low_on_hand_profile,
low_on_hand_profile(Date_Range), low_on_hand(Date), low_on_hand
(Date_Range), low_on_hand(Date_Range, Logical).

Extension BASIC_FILTER_AND_RANK

The Fields in this Extension are
min_time, min_on_hand, excess_on_hand, producing_operation,
supplying_operation, flow_plan_selection_start, flow_plan_selection_end,
flow_plan_selection_interval, flow_plan_filter, flow_plan_rank,
continue_flow_plan_selection, flow_plan_resize_quantity_range,
flow_plan_move_restriction, flow_plan_split_restriction.

Extension BASIC_FILTER_AND_RANK

The Fields in this Extension are
max_target_profile, max_target_profile(Date_Range), max_target(Date),
max_target(Date_Range), max_target(Date_Range, Logical), safety_target_profile,
safety_target_profile(Date_Range), safety_target(Date), safety_target
(Date_Range), safety_target(Date_Range, Logical), cycle_on_hand_profile,
cycle_on_hand_profile(Date_Range), cycle_on_hand(Date), cycle_on_hand
(Date_Range), cycle_on_hand(Date_Range, Logical), excess_on_hand_profile,
excess_on_hand_profile(Date_Range), excess_on_hand(Date), excess_on_hand
(Date_Range), excess_on_hand(Date_Range, Logical), low_on_hand_profile,
low_on_hand_profile(Date_Range), low_on_hand(Date), low_on_hand
(Date_Range), low_on_hand(Date_Range, Logical).

Extension FIXED_QUANTITY

The Fields in this Extension are
min_on_hand, min_time, excess_on_hand, quantity, producing_operation,
supplying_operation.

Extension FIXED_QUANTITY

The Fields in this Extension are
max_target_profile, max_target_profile(Date_Range), max_target(Date),
max_target(Date_Range), max_target(Date_Range, Logical), safety_target_profile,
safety_target_profile(Date_Range), safety_target(Date), safety_target
(Date_Range), safety_target(Date_Range, Logical), cycle_on_hand_profile,
cycle_on_hand_profile(Date_Range), cycle_on_hand(Date), cycle_on_hand

(Date_Range), cycle_on_hand (Date_Range, Logical), excess_on_hand_profile, excess_on_hand_profile (Date_Range), excess_on_hand (Date), excess_on_hand (Date_Range), excess_on_hand (Date_Range, Logical), low_on_hand_profile, low_on_hand_profile (Date_Range), low_on_hand (Date), low_on_hand (Date_Range), low_on_hand (Date_Range, Logical).

Extension MULTIPLE

The fields in this Extension are
min_on_hand, excess_on_hand, quantity_range, multiple_quantity, min_time, producing_operation, supplying_operation.

Extension MULTIPLE

The fields in this Extension are
max_target_profile, max_target_profile (Date_Range), max_target (Date), max_target (Date_Range), max_target (Date_Range, Logical), safety_target_profile, safety_target_profile (Date_Range), safety_target (Date), safety_target (Date_Range), safety_target (Date_Range, Logical), cycle_on_hand_profile, cycle_on_hand_profile (Date_Range), cycle_on_hand (Date), cycle_on_hand (Date_Range), cycle_on_hand (Date_Range, Logical), excess_on_hand_profile, excess_on_hand_profile (Date_Range), excess_on_hand (Date), excess_on_hand (Date_Range), excess_on_hand (Date_Range, Logical), low_on_hand_profile, low_on_hand_profile (Date_Range), low_on_hand (Date), low_on_hand (Date_Range), low_on_hand (Date_Range, Logical).

Extension MULTIPLE_FILTER_AND_RANK

The fields in this Extension are
min_on_hand, excess_on_hand, quantity_range, multiple_quantity, min_time, producing_operation, supplying_operation, flow_plan_selection_start, flow_plan_selection_end, flow_plan_selection_interval, flow_plan_filter, flow_plan_rank, continue_flow_plan_selection, flow_plan_resize_quantity_range, flow_plan_move_restriction, flow_plan_split_restriction.

Extension MULTIPLE_FILTER_AND_RANK

The fields in this Extension are
max_target_profile, max_target_profile (Date_Range), max_target (Date), max_target (Date_Range), max_target (Date_Range, Logical), safety_target_profile, safety_target_profile (Date_Range), safety_target (Date), safety_target (Date_Range), safety_target (Date_Range, Logical), cycle_on_hand_profile, cycle_on_hand_profile (Date_Range), cycle_on_hand (Date), cycle_on_hand (Date_Range), cycle_on_hand (Date_Range, Logical).

(Date_Range), cycle_on_hand (Date_Range, Logical), excess_on_hand_profile, excess_on_hand_profile (Date_Range), excess_on_hand (Date), excess_on_hand (Date_Range), excess_on_hand (Date_Range, Logical), low_on_hand_profile, low_on_hand_profile (Date_Range), low_on_hand (Date), low_on_hand (Date_Range), low_on_hand (Date_Range, Logical).

Extension FIXED_QUANTITY_FENCED

The fields in this Extension are
min_on_hand, min_time, excess_on_hand, quantity, fence, after_fence_excess_on_hand, after_fence_quantity, producing_operation, supplying_operation.

Extension FIXED_QUANTITY_FENCED

The fields in this Extension are
max_target_profile, max_target_profile (Date_Range), max_target (Date), max_target (Date_Range), max_target (Date_Range, Logical), safety_target_profile, safety_target_profile (Date_Range), safety_target (Date), safety_target (Date_Range), safety_target (Date_Range, Logical), cycle_on_hand_profile, cycle_on_hand_profile (Date_Range), cycle_on_hand (Date), cycle_on_hand (Date_Range), cycle_on_hand (Date_Range, Logical), excess_on_hand_profile, excess_on_hand_profile (Date_Range), excess_on_hand (Date), excess_on_hand (Date_Range), excess_on_hand (Date_Range, Logical), low_on_hand_profile, low_on_hand_profile (Date_Range), low_on_hand (Date), low_on_hand (Date_Range), low_on_hand (Date_Range, Logical).

Extension FIXED_TIME

The fields in this Extension are
produce_time, min_on_hand, min_time, excess_on_hand, producing_operation, supplying_operation, supply_time.

Extension FIXED_TIME

The fields in this Extension are
max_target_profile, max_target_profile (Date_Range), max_target (Date), max_target (Date_Range), max_target (Date_Range, Logical), safety_target_profile, safety_target_profile (Date_Range), safety_target (Date), safety_target (Date_Range), safety_target (Date_Range, Logical), cycle_on_hand_profile, cycle_on_hand_profile (Date_Range), cycle_on_hand (Date), cycle_on_hand (Date_Range), cycle_on_hand (Date_Range, Logical), excess_on_hand_profile, excess_on_hand_profile (Date_Range), excess_on_hand (Date), excess_on_hand (Date_Range), excess_on_hand (Date_Range, Logical), low_on_hand_profile, low_on_hand_profile (Date_Range), low_on_hand (Date), low_on_hand (Date_Range), low_on_hand (Date_Range, Logical).

Summary Section	Wall ()
-----------------	----------

Extension STANDARD

The Fields in this Extension are

Extension STANDARD

The Fields in this Extension are

Extension CUSTOM

The Fields in this Extension are

Extension CUSTOM

The Fields in this Extension are

Extension LFL_SIMPLE

The Fields in this Extension are
producing_operation, consuming_operation, supplying_operation.

Extension LFL_BOUNDED

The Fields in this Extension are
quantity_range, rough_fence, rough_quantity_range, producing_operation, consuming_operation, supplying_operation.

Extension MLFL_BOUNDED

The Fields in this Extension are
quantity_range, rough_fence, rough_quantity_range, producing_operation, consuming_operation, supplying_operation.

Extension MANUAL

The Fields in this Extension are

Extension CALENDAR

The Fields in this Extension are
units_of_safety_stock, user_bias, customer_service_level, default_mean_demand, default_mean_demand_time_bucket, mean_demand_calendar, default_standard_deviation_demand, standard_deviation_demand_calendar, default_mean_lead_time, mean_lead_time_calendar, default_standard_deviation_lead_time, standard_deviation_lead_time_calendar, safety_factor, safety_factor(Date), demand_safety_stock, demand_safety_stock(Date), supply_safety_stock, supply_safety_stock(Date).

Extension NUMBER

Summary Section	Wall ()
-----------------	----------

The Fields in this Extension are
before_horizon_number, after_horizon_number, intra_horizon_number.

Extension QUANTITY

The Fields in this Extension are
before_horizon_quantity, after_horizon_quantity, intra_horizon_quantity.

Extension NUMBER_QUANTITY

The Fields in this Extension are
before_horizon_number, after_horizon_number, intra_horizon_number, before_horizon_quantity, after_horizon_quantity, intra_horizon_quantity.

Extension SYMBOL

The Fields in this Extension are
before_horizon_symbol, after_horizon_symbol, intra_horizon_symbol.

Extension TIME

The Fields in this Extension are
before_horizon_time, after_horizon_time, intra_horizon_time.

Extension EVERYDAY

The Fields in this Extension are

Extension EVERY_N_DAYS

The Fields in this Extension are
nth.

Extension WEEKDAYS

The Fields in this Extension are

Extension WEEKENDS

The Fields in this Extension are

Extension DAYS_OF_WEEK

The Fields in this Extension are
days.

Extension DAY_OF_MONTH

The Fields in this Extension are
day, months.

Summary Section	wait ()
-----------------	---------

Extension DAY_OF_WEEK_OF_MONTH
The Fields in this Extension are
day, nth, week, months.

Extension DAY_OF_LAST_WEEK_OF_MONTH
The Fields in this Extension are
day, months.

Extension DAY_OF_YEAR
The Fields in this Extension are
day.

Extension YEARLY
The Fields in this Extension are
month, day.

Extension NUMBER
The Fields in this Extension are
number.

Extension QUANTITY
The Fields in this Extension are
quantity.

Extension NUMBER_QUANTITY
The Fields in this Extension are
number, quantity.

Extension SYMBOL
The Fields in this Extension are
symbol.

Extension TIME
The Fields in this Extension are
time.

Extension VOID
The Fields in this Extension are
specification.

Extension Logical

Summary Section	wait ()
-----------------	---------

The Fields in this Extension are
specification.

Extension String
The Fields in this Extension are
specification.

Extension Symbol
The Fields in this Extension are
specification.

Extension Date
The Fields in this Extension are
specification.

Extension Date_Range
The Fields in this Extension are
specification, separator.

Extension Number
The Fields in this Extension are
specification.

Extension Percentage
The Fields in this Extension are
specification.

Extension Integer
The Fields in this Extension are
specification.

Extension Quantity
The Fields in this Extension are
specification.

Extension Quantity_Range
The Fields in this Extension are
specification, separator.

Extension Time
The Fields in this Extension are

Summary Section	wait()
-----------------	--------

specification.

Extension Restriction

The Fields in this Extension are specification.

Extension Horizon_Date

The Fields in this Extension are
time_format, number_format, weekday_format, day_format, rel_day_of_month_format, day_of_month_format, day_from_end_month_format, DOW_format, DOW_of_week_format, DOW_of_month_format, DOW_of_week_month_format.

Extension List

The Fields in this Extension are
delimiter, element_format, width, truncated_format.

Extension SIMPLE

The Fields in this Extension are

Extension SELECTOR

The Fields in this Extension are

Extension EXTENDED

The Fields in this Extension are

Extension USER

The Fields in this Extension are
get_expression, set_expression.

Modelle	wait()
---------	--------

5 Models

Supply_Chain

Site

LINK

SUPPLIER

CUSTOMER

Location

Item

Buffer

Buffer_Problem_Detector

Flow_Criterion

Resource

Resource_Skill

Size_Dimension

Resource_Setup_Order

Resource_Blocks

Skill

Skill_Resource

Operation

Load

Load_Size

Flow

Operation_Problem_Detector

Alternate_Operation

Effective_Calendar_Operation

Routing_Operation

Configuration

Item_Group

Sub_Item_Group

Sub_Item

Seller

Site_Group

Explicit_Site

Product_Root

Product_Supplier

Product_Item

Models	wait()
--------	--------

- Product
 - Generic_Product
 - Alternate_Product
 - Product_Allocation
 - Product_Group
 - Sub_Product_Group
 - Sub_Product
- Plan
 - Site_Plan
 - LINK
 - SUPPLIER
 - CUSTOMER
 - Buffer_Plan
 - Lot
 - Sorted_Bucket
 - Resource_Plan
 - Consolidation
 - Operation_Plan
 - Load_Plan
 - Flow_Plan
 - Lot_Flow
 - Operation_State
 - Request
 - Delivery_Request
 - Item_Request
 - Promise
 - Delivery_Promise
 - Delivery_Available_To_Promise
 - Item_Promise
 - Item_Available_To_Promise
 - Product_Available_To_Promise
 - Acceptance
 - Delivery_Acceptance
 - Item_Acceptance
 - Seller_Plan
 - Forecast
 - INDIVIDUAL

Models	wait()
--------	--------

- GROUP
 - Forecast_Entry
 - ATP_Entry
- Problem
 - Active_Strategy
 - Active_Problem
 - Active_Goal
- Problem_Category
- Horizon
 - Horizon_Bucket_Start
- Strategy
 - Problem_Set
 - Strategy_Change
 - Strategy_Lock
 - Strategy_Goal
- Ordered_Sub_Strategy
- Ranked_Sub_Strategy
- Unit
 - Unit_Quantity
 - Profile_Number
 - Profile_Percentage
 - Profile_Quantity
 - Box
 - Calendar_Entry
 - Calendar
 - Sub_Calendar
 - Calendar_Plan
 - Worksheet
 - Control
 - Connection
 - User
 - Report_Directory
 - Report
 - Layout
 - Style
 - Format
 - Domain

Models	wait()
--------	--------

Model_Type
Field
Extension_Selector
Field_Error
Function
Breakpoint

Models	Supply_Chain Model
--------	--------------------

5.1 Supply_Chain Model

Supply_Chain -- *an independent (top-level) model*

Supply_Chain Definition

The Supply_Chain models a set of Sites (organizational units) that make up a supply chain to be managed and planned. The Sites that make up a Supply_Chain can be modeled in detail, or may be modeled as a "black box". Detailed site models will model the material flow through the site in great detail by modeling operations, resources, buffers. This will also allow detailed planning of demand placed on them. On the other hand, a "black box" site will only model the requests for manufactured/supplied items or promises to supply those items.

Several Supply_Chains may be modeled at once. This capability is primarily used for What-If analysis of changes to a Supply_Chain but it can be also used to model multiple concurrent and independent supply chains within one RHYTHM engine.

Example

The existing supply chain network has distribution centers in Boston, Chicago, Dallas and Denver. With the capability of modeling multiple supply chains, you could create a scenario where you can add a distribution center in Los Angeles for the west coast customers and plan the same requests to see what the impact of opening a new distribution center would be on your plan.

Plan Definition

Each Supply_Chain may have one or more Plans. A Plan models what a Supply_Chain is currently doing, its current condition, and what it plans to do in the future. It holds the "live" data, whereas the Supply_Chain holds the "master" data. Typically, a Supply_Chain will have only one Plan. However, What-If analysis on Plans can be very important. In particular, there will typically be one active or "released" Plan and another What-If Plan built from the "released" plan for deciding what to release next.

What-If plans can also model different scenarios of forecasted demand and product mix on the same supply chain.

Each Plan may be simulated. There may be several simulations of each Plan using different stochastic data. (Stochastic simulations are not currently supported.)

The Supply_Chain model has these submodels :
Site, Seller, Plan.

The Supply_Chain model has fields that references these models :
Site, Seller, Plan.

These models have a field that is a Supply_Chain model :
Site, Seller, Plan.

The key field for this model is name
This model may be extended with user-defined fields.

name - - a Symbol field of model Supply_Chain
name Field The name of this Supply_Chain.
Default: None -- this is a key field

description - - a String field of model Supply_Chain
description Field A description of this Supply_Chain. For example, it may be a description of what is being explored in this What-If.
Default: none

sites - - a list of Site submodels of model Supply_Chain
sites Field The Sites (organizational units) that make up this Supply_Chain. All demand (Requests and Promises) is placed between Sites. The activity within a Site may be modeled or not. When modeled, it may be modeled at various levels of detail.

top_sites - - a List(Site) field of model Supply_Chain
top_sites Field A subset of 'sites' that only includes Sites whose 'organization' is [unspecified]. These Sites are the topmost organizations; they are not within another organization. This list is particularly useful as a starting point for displaying the hierarchy of Sites.
Properties: Export-Only Field

sellers - - a list of Seller submodels of model Supply_Chain
sellers Field The sales personnel and sales organizations responsible for forecasting and selling the Products of this Supply_Chain. A single sales organization may sell Products for one or many of the Sites in the Supply_Chain. Each Seller consists of a forecast for each of the Products it sells. Based upon those forecasts, demand is placed upon the Sites, production and distribution plans are created, and the resultant supply is allocated to the Sellers for promising to the customers as the actual demand arrives.

top_sellers - - a List(Seller) field of model Supply_Chain
top_sellers Field A subset of 'sellers' that only includes Sellers whose 'organization' is [unspecified]. These Sellers are the topmost organizations; they are not within another organization. This list is particularly useful as a starting point for displaying the hierarchy of Sellers.
Properties: Export-Only Field

plans - - a list of Plan submodels of model Supply_Chain
plans Field The various Plans that have been built for managing the 'owner' Supply_Chain. Each Plan contains a Site_Plan for each Site and a Seller_Plan for each Seller in this Supply_Chain. The various plans can represent alternative scenarios and alternative ways to respond to those scenarios. In that way, several "What If..." Plans can be created, manipulated, and compared intelligently.

Models	Site Model
--------	------------

5.1.1 Site Model

Site -- a submodel of model Supply_Chain

The Site models an organizational unit to be planned. Each Site has independent namespace, authority, privacy, etcetera. Note that a Site could correspond to a plant, to a portion of a plant, to several plants, or to a plant, warehouses, distribution centers, and stores. The Site is not a physical division, but rather an organizational one. It defines a portion of the Supply_Chain that is planned and controlled by one team of decision makers.

Note however that a team of decision makers is not limited to one Site. Several Sites may all be designated as 'managed' by a single RHYTHM SCP engine. Thus, if a single team of planners has authority over two plants that need separate namespaces (e.g., they use the same Resource names to mean different things), then an independent Site can be setup for each plant, but both Sites will have 'managed' set 'true'.

Demand between 'managed' and non-'managed' Sites is placed formally as Requests, for which they receive Promises back. The Requests and Promises embody agreements between those separately managed organizations. The promising Site makes Plans to fulfill those Promises. The requesting Site makes Plans assuming those Promises will be fulfilled.

In contrast, demand within a Site and demand between 'managed' Sites is propagated directly and transparently, since it is all under the control of one team of decision makers. All of the elements of the 'managed' Sites are planned as an integrated whole, as a single unit.

Fundamentally, a Site consists of Operations, Resources, and Buffers, which together form the FLO (Flow-Load-Operation) network. Each Site's FLO network is independent (no shared Resources or Buffers).

The FLO network representation is designed to integrate material and capacity planning, to integrate master and execution planning, and to integrate factory and distribution planning. It is the foundation for achieving Truly Integrated Planning. Furthermore, the FLO network was designed to provide adequate extensibility such that it can be optimally suited for each part of the supply chain. Most supply chains consist of Sites with very different needs (a steel factory feeds a repetitive bearings maker that feeds a specialized products maker which feeds a consumer products maker which feeds a warehouse which feeds a distribution center which feeds a retailer). A software system designed to integrate the supply chain planning should be the best tool for each Site, as well as the best at the inter-site issues.

Models	Site Model
--------	------------

Note that if you are looking for the traditional routing model, look in Operation. The Operation model is flexible and powerful enough to model routings as well as steps in routings. Similarly, the Operation model covers transform and movement of Items, and thus covers Bills-of-Materials and Bills-of-Distribution -- see Operation and, in particular, the Flow submodel within Operation.

If you are looking for a representation of SKUs (stockkeeping units), look at Buffer (which models the flow of an Item at a Location). In fact, many of the characteristics found in traditional systems' Item Master files are moved from the Item model to the Buffer model, allowing variation at different locations (critical for integrated supply chain support).

The "live" and "planning" data is held in the corresponding models suffixed with "_Plan". For example, the Resource_Plan models the planned load, maintenance, setup, etc., of a Resource. The Buffer_Plan models the planned flow into and out of a Buffer. The Operation_Plan models the state and plans for Operations to be performed. See the Site_Plan model for more detail on the "_Plan" models corresponding to the Site's models.

Note that certain naming conventions have been strictly followed to increase consistency and reduce ambiguity. For example, within the FLO Network within a Site, the terms "produce" and "consume" are used. An Operation consumes from some Buffers and produces into other Buffers. Of course, an Operation may produce into a Buffer without building anything, it may just move the items, or even just re-grade them (use a 150MHz chip as a 100MHz one). Outside of the Site, for the inter-site relationships, the terms "supply" and "purchase" (or "supplier" and "customer") are used. Thus, you will have "supplying Sites" and "producing Operations", but never the other way around.

The model has selectors:
role,

The Site model has fields that references these models:
Site, Site_Plan, Supply_Chain.

These models have a field that is a Site model:
Supply_Chain, Site, Operation, Resource, Buffer, Product_Root, Product, Site_Plan, Location, Item, Skill, Configuration, Item_Group, Explicit_Site, Product_Supplier, REQUEST_FIXED, REQUEST_FIXED_WITH_ANALYSIS, REQUEST_FIXED_ALTERNATES.

The key field for this model is name.
This model may be extended with user-defined fields.

name -- *a Symbol field of model Site*

The name of this Site.

Default: None -- this is a key field

description -- *a String field of model Site*

A description of this Site.

Default: none

category -- *a Symbol field of model Site*

A simple user-defined categorization of Sites. This is often used in reports. This will allow users to filter the Sites by category, and to write expressions that depend upon the category of the Site.

The standard reports look for the values "Manufacturing" and "Distribution".
Default: Manufacturing

sub_category -- *a Symbol field of model Site*

A simple user-defined categorization of Sites. This is often used in reports. This will allow users to filter the Sites by category, and to write expressions that depend upon the category of the Sites.

The standard reports do not look for any specific values. It is commonly used to categorize by level or function within the Supply Chain. For example, in Distribution, this may be "Retail", "Customer DC", "Regional DC", "Warehouse", or "Packaging". But any values can be used -- whatever is useful for categorization in reports.

Default: [unspecified]

rank -- *a Number field of model Site*

Rank of this site. This is used to compute the planning priority of an item_rmp in active_strategy.

Default: 0

organization -- *a Site field of model Site*

The organization Site of which this Site is a member. 'organization' and 'members' these two fields together provides ability to model hierarchical site organization.
Default: [unspecified]

members -- *a List(Site) field of model Site*
The Sites which specify this Site as their 'organization'.
Properties: Export-Only Field

member_of (Explicit_Site) -- *a Logical field of model Site*

Returns "true" if this Site or any of its 'organization's is the parameter Site. More precisely, if this Site is the parameter Seller, then it returns "true"; otherwise, if this Site's 'organization' is [unspecified], then it returns "false"; otherwise, it returns 'organization.member_of(parameter)'.
Properties: Export-Only Field

role -- *an extension selector of model Site*

Defines the role this Site plays in the Supply_Chain: SUPPLIER, LINK, or CUSTOMER. The LINK Sites of the Supply_Chain transform items obtained from SUPPLIER Sites into the items requested by CUSTOMER Sites. The SUPPLIER Sites introduce material into the Supply_Chain. The CUSTOMER Sites remove material from the Supply_Chain. The LINK Sites transform and move material within the Supply_Chain.

LINK Sites can be modeled in detail. They consist of Operations which use Resources to transform or transfer items between Buffers (the FLO network). They place Requests on SUPPLIER and other LINK Sites, and provide Promises to CUSTOMER and other LINK Sites.

CUSTOMER Sites are simply the delivery destinations of LINK Sites. CUSTOMER Sites place Requests upon LINK Sites and receive back Promises.

SUPPLIER Sites are similarly the sources of Promises for the Requests from the LINK Sites.

Default: CUSTOMER

Extensions:

LINK, SUPPLIER, CUSTOMER.

margin_target (Date, Date) -- *a Money field of model Site*

The desired difference between the price received for Products and the marginal cost of producing those items during the given Date_Range (from start Date to end Date). The purpose of this field is to account for the fixed costs involved with having, maintaining, and running this Site. A profit that covers the marginal costs is not sufficient. Thus, the difference between the price and the marginal costs (the margin) must be large enough to cover the fixed costs and result in a strategically desirable profit. This is computed in strategic planning and is an input to master and operational planning.

THIS IS JUST A SKETCH, as are all cost and margin fields currently.
Properties: Export-Only Field

delivery_name - - - *a String field of model Site*
The name of the customer for delivery.

Default: none

delivery_contact - - - *a String field of model Site*
The name of the individual to contact concerning deliveries.

Default: none

delivery_phone - - - *a String field of model Site*
The phone number for delivery.

Default: none

delivery_fax - - - *a String field of model Site*
The fax phone number for delivery.

Default: none

delivery_address - - - *a String field of model Site*
The street address for delivery.

Default: none

delivery_city - - - *a String field of model Site*
The city for delivery.

Default: none

delivery_state - - - *a String field of model Site*
The state or province for delivery.

Default: none

delivery_country - - - *a String field of model Site*
The country for delivery.

Default: none

delivery_postal_code - - - *a String field of model Site*
The postal code for delivery.

Default: none

billing_name - - - *a String field of model Site*
The contact name for billing.

Default: none

billing_contact - - - *a String field of model Site*
The name of the individual to contact for billing.

Default: none

billing_phone - - - *a String field of model Site*
The phone number for billing.

Default: none

billing_fax - - - *a String field of model Site*
The fax phone number for billing.

Default: none

billing_address - - - *a String field of model Site*
The street address for billing.

Default: none

billing_city - - - *a String field of model Site*
The city for billing.

Default: none

billing_state - - - *a String field of model Site*
The state for billing.

Default: none

billing_country - - - *a String field of model Site*
The country for billing.

Default: none

billing_postal_code - - - *a String field of model Site*
The postal code for billing.

Default: none

tax_identification - - - *a String field of model Site*
The Customer's tax identification number for reduced tax.

Default: none

site_plan (Plan) - - - *a Site_Plan field of model Site*
The Site_Plan for this Site in the specified Plan.
Properties: Export-Only Field

Models	Site Model
--------	------------

owner -- a Supply_Chain field of model Site

Properties: Export-Only Field

Models	Standard Extensions of model Site
--------	-----------------------------------

5.1.1.1 Standard Extensions of model Site

5.1.1.1.1 role extensions of model Site

5.1.1.1.1

LINK -- a role extension of model Site

A LINK Site is a "link" in this supply "chain". It purchases Items from zero or more other Sites, performs Operations on Items, moving or transforming them, and supplies Items to zero or more other Sites.

LINK Sites can be modeled in detail. They consist of Operations which use Resources to transform or transfer Items between Buffers (the FLO network). They place Requests on other LINK and SUPPLIER Sites, and provide Promises to other LINK and CUSTOMER Sites.

The LINK model has these submodels :

Location, Item, Buffer, Resource, Skill, Operation, Configuration, Item_Group.

The LINK model has fields that reference these models :

Location, Item, Buffer, Resource, Skill, Operation, Configuration, Item_Group.

managed -- a Logical field of model LINK

If "true", then this LINK Site is planned and managed by this model. The plans created for such a Site are assumed to be the plans it will follow.

If "false", then this LINK Site is managed elsewhere and this model is simply modeling its behavior. Such modeling is purely informational -- the plans created for such a Site cannot be assumed to be reliable. Only the Promises provided from the managers of that Site can be relied upon (hopefully, anyway).

Requests and Promises will be created between all unmanaged Sites and the managed Sites, just as would be created with SUPPLIER Sites and CUSTOMER Sites. In contrast, managed Sites will interface directly to each other. There is no sense in a planner making Requests to himself or responding with formal Promises to himself. Requests and Promises will be created for informational and reporting purposes, but they will be maintained automatically -- as if there was no Site boundary at all.

Default: true

locations -- a list of Location submodels of model LINK

The Locations within this Site.

Models	LINK Extension
--------	----------------

top_locations -- *a List(Location) field of model LINK*
 The Locations within this Site whose super_location is [unspecified]. These are the topmost Locations at this Site. This is a useful starting point for generating a Location hierarchy.
 Properties: Export-Only Field

items -- *a list of Item submodels of model LINK*
 The Items consumed, used, and/or produced at this Site. Requests placed on this Site must be for Items in this list that are scellable.

top_items -- *a List(Item) field of model LINK*
 The Items within this Site whose 'family' is [unspecified]. These are the topmost Locations at this Site. This is a useful starting point for generating an Item hierarchy.
 Properties: Export-Only Field

buffers -- *a list of Buffer submodels of model LINK*
 The Buffers to be used for managing the flow of Items at this Site.

resources -- *a list of Resource submodels of model LINK*
 The Resources which model the capacity to perform Operations at this Site.
skills -- *a list of Skill submodels of model LINK*
 The Skills needed to perform Operations; Resources have these Skills.

operations -- *a list of Operation submodels of model LINK*
 The Operations that transform Items into other Items and/or move Items between Buffers.

configurations -- *a list of Configuration submodels of model LINK*
 Should be hidden. The list of these is not relevant.

item_groups -- *a list of Item_Group submodels of model LINK*
 Item_Groups group items into hierarchies. Each item can appear in at most one Item_Group in a hierarchy of Item_Groups. But each item can appear in any number of independent Item_Group hierarchies.

top_item_groups -- *a List(Item_Group) field of model LINK*
 This List is a subset of item_groups consisting only of Item_Groups that are the top of a Item_Group hierarchy, the Item_Groups that do not appear within any other Item_Group.
 Properties: Export-Only Field

Models	SUPPLIER Extension
--------	--------------------

buffers_at_level(Integer) -- *a List(Buffer) field of model LINK*
 A List of the Buffers at a particular Integer 'level'. Note, 'site.buffers_at_level(3)' is equivalent to 'site.buffers.filter(#level == 3)'; but is likely less expensive to compute.
 Properties: Export-Only Field

operations_at_level(Integer) -- *a List(Operation) field of model LINK*
 A List of the Operations at a particular Integer 'level'. An operation level is defined as the min buffer level of all its consuming flow buffers.
 Properties: Export-Only Field

buffer_levels -- *a List(Integer) field of model LINK*
 A List of all the level Integer values used by any of 'buffers'. It is a List containing 0, and the each Integer in order up to the largest 'level' code of any of 'buffers'. This List is generally used to feed the 'buffers_at_level' function in order to generate a table of the Buffers by-level.
 Properties: Export-Only Field

operation_levels -- *a List(Integer) field of model LINK*
 A List of all the level Integer values used by any of 'operations'. This List is generally used to feed the 'operations_at_level' function in order to generate a table of the Operations by-level.
 Properties: Export-Only Field

5.1.1.1.2 SUPPLIER -- a role extension of model Site

A SUPPLIER Site is not planned or modeled in detail; rather it represents the Items that can be procured from a supplier by LINK Sites, the Requests for those Items, and the Promises received from the supplier.

The SUPPLIER model has these submodels :

Item.

The SUPPLIER model has fields that references these models :

Item.

items -- *a list of Item submodels of model SUPPLIER*
 The Items produced at this Site. Requests placed on this Site must be for one of these Items.

Models	CUSTOMER Extension
--------	--------------------

5.1.1.1.3 CUSTOMER -- a role extension of model Site

A CUSTOMER Site is not planned or modeled in detail; rather, it is simply a destination for deliveries and source of Requests.

Models	role submodels of model Site
--------	------------------------------

5.1.1.2 role submodels of model Site 5.1.1.3 Location Model

Location -- a submodel of model Site

A physical location within a surrounding "super" Location. The 'sub_locations' are considered 'within' their 'super_location'. Thus, the Locations form a hierarchy which can be arbitrarily deep.

The Location's coordinates are relative to the others within the same 'super_location'. The 'super_location' and all its 'sub_locations' are placed as a group according to the 'super_location's coordinates. Note that the coordinates are purely informational, to support graphical Maps of the Locations. The coordinates have no effect on planning. See TRANSIT Operations for computation of transit Time.

In each Site there is one special uneditable Location named [unspecified], which indicates no Location has been specified. This is typically the default value for Location fields. It is treated as if it is everywhere, anywhere, or nowhere, as is appropriate.

The Location model has fields that references these models :
Location, Box, Site.

These models have a field that is a Location model :
Resource, Resource_Plan, Buffer, LINK, Location, Load, TRANSIT.

The key field for this model is name
This model may be extended with user-defined fields.

name -- a Symbol field of model Location
The name of this Location. This name must be unique among Locations at this Site or empty (unnamed).

Default: None -- this is a key field

description -- a String field of model Location
A description of this Location.

Default: none

category -- a Symbol field of model Location

A simple user-defined categorization of Locations. This is often used in reports. This will allow users to filter the Locations by category, and to write expressions that depend upon the category of the Location.

Models	Location Model
--------	----------------

The standard reports look for the values "Manufacturing", "Distribution", and "Inventory".

Default: Manufacturing

sub_category -- *a Symbol field of model Location*

A simple user-defined categorization of Locations. This is often used in reports. This will allow users to filter the Locations by category, and to write expressions that depend upon the category of the Locations.

The standard reports don not look for any specific values. It is commonly used to categorize by mode, method, duration, cost, or usage pattern. For example, in Distribution, this may be "Dock", "Receiving", "Staging", "Shelf", or "Lane". But any values can be used -- whatever is useful for categorization in reports.

Default: [unspecified]

super_location -- *a Location field of model Location*

The Location inside of which this one is located. All the coordinates of this Location are relative to the inside of its super_location.

Default: [unspecified]

sub_locations -- *a List(Location) field of model Location*

The Locations that are immediately within this Location. The Locations that specify this Location as their super_location.

Properties: Export-Only Field

within (Location) -- *a Logical field of model Location*

Returns "true" if this Location is within the argument Location. It will return "true" if passed its super_location, or its super_location's super_location, and so on up to and including [unspecified].

For example, if 'dock' is Location "Loading Dock 1-3" which is in Building One, and building is Location "Building One", then 'dock.within(building)' returns "true", whereas building.within(dock) returns "false". Note that all Locations are within "[unspecified]", and "[unspecified]" is within all other Locations.

Properties: Export-Only Field

box -- *a Box field of model Location*

box of this Location within the super_location. Note that this is purely informational, to support graphical Map's of the Locations. Its value has no effect on planning. See TRANSIT Operations for computation of transit Time.

Default: unspecified

Models	Location Model
--------	----------------

Properties: command=True

x_position -- *a Quantity field of model Location*

X-Coordinate of this Location within the super_location. Note that this is purely informational, to support graphical Map's of the Locations. Its value has no effect on planning. See TRANSIT Operations for computation of transit Time.

Default: 0

Properties: command=True

x_size_min -- *a Quantity field of model Location*

The minimum size of this Location along the X-axis. If Locations within this one have larger x_position's, then x_size will be larger. That is, every Location is large enough to hold all the Location positions within it. This allows a minimum size to be specified when the Location is larger than indicated by its children (it may not even have children).

Default: 0

Properties: command=True

x_size -- *a Quantity field of model Location*

The size of this Location along the X-axis. Note that this is purely informational, to support graphical Map's of the Locations. Its value has no effect on planning. See TRANSIT Operations for computation of transit Time.

Properties: command=True Export-Only Field

y_position -- *a Quantity field of model Location*

Y-Coordinate of this Location within the super_location. Note that this is purely informational, to support graphical Map's of the Locations. Its value has no effect on planning. See TRANSIT Operations for computation of transit Time.

Default: 0

Properties: command=True

y_size_min -- *a Quantity field of model Location*

The minimum size of this Location along the Y-axis. If Locations within this one have larger y_position's, then y_size will be larger. That is, every Location is large enough to hold all the Location positions within it. This allows a minimum size to be specified when the Location is larger than indicated by its children (it may not even have children).

Default: 0

Properties: command=True

Models	Item Model
--------	------------

y, size - - - *a Quantity field of model Location*
The size of this Location along the Y-axis. Note that this is purely informational, to support graphical Maps of the Locations. Its value has no effect on planning. See TRANSIT Operations for computation of transit Time.
Properties: **command**=True **Export-Only Field**

owner - - - *a Site field of model Location*

Properties: **Export-Only Field**

5.1.1.4 Item Model

Item - - - *a submodel of model Site*

An Item models a kind of material, part, component, subassembly, assembly, or good which has particular characteristics that define how it can be built, stored, processed, or used.

Note that some of the traditional fields of Item have been split out into the Buffer model. This is important for supply chain modeling and for integrated material and capacity planning. The management of an Item at a particular Location (a SKU) is modeled by Buffer. Any Location-specific or SKU-specific information of an Item is modeled by Buffer.

Further, any pricing, availability, or forecasting information is split out into the Product model. This is done because modern pricing and marketing programs involve much more than just the physical Item -- the Customer and the Order Lead Time may also be at issue.

In each Site, there is a special uneditable Item named [unspecified]. It has the default values for all fields.

The model has selectors:
spec.

The Item model has fields that references these models :
Item, Operation, Unit, Site.

These models have a field that is a Item model :

Models	Item Model
--------	------------

Buffer, LINK, Item, Configuration, SUPPLIER, Product_Item, Item_Request, Item_Acceptance, Item_Promise, Sub_Item, DELIVER, RECEIVE, STORAGE, REQUEST_FIXED, REQUEST_FIXED_WITH_ANALYSIS, REQUEST_FIXED_ALTERNATES, STARTED, COMPLETED, Feature.

The key field for this model is name
This model may be extended with user-defined fields.

name - - - *a Symbol field of model Item*

The name of this Item. This 'name' must be unique among Items at this Site.
Default: None -- this is a key field

description - - - *a String field of model Item*

A description of this Item.
Default: none

category - - - *a Symbol field of model Item*

A simple user-defined categorization of Items. This is often used in reports. This will allow users to filter the Items by category, and to write expressions that depend upon the category of the Items.

The standard reports look for the values "Purchased", "Intermediate", "End", "Plantion", "Resource", and "Cash". The first four are the standard APICS definitions (though there are common synonyms: "Raw Material" for "Purchased" Items, "WIP" for "Intermediate" Items, and "Finished Goods" for "End" Items). "Resource" Items are typically either renewable Resources (e.g., drill bits), Resources that travel with the material (e.g., molds, dies, mandril), or Resources that are flow-limited rather than load-limited (e.g., subcontractor that commits to a certain quantity). A "Cash" Item can be used to model cash flow.

Default: End

drawing_id - - - *a Symbol field of model Item*

An identifier for a drawing or graphic of this Item.
Default: none

family - - - *a Item field of model Item*

The Item family to which this Item belongs. An Item family is generally an 'artificial' Item that is used for categorization and grouping. It is useful in generating Reports that include numerous related Items. It does not otherwise effect planning computations. If [unspecified], then this Item is a 'top_item' of the 'owner Site'.
Default: [unspecified]

Models	Item Model
--------	------------

children - - *a List(Item) field of model Item*
 This List contains each Item that specifies this Item as its 'family'. If this list is empty then it is not an Item family.
 Properties: Export-Only Field

artificial - - *a Logical field of model Item*
 If "true", then this Item is not real -- it cannot be built, purchased, or obtained -- it does not really exist. For example, an Item family often an artificial Item, it just represents some common characteristics of a family of Items.
 Default: true

spec - - *an extension selector of model Item*
 The spec extension defines the additional specification fields required in a Configuration of this Item. It defines interchangeability of these Items.

For instance, an **OPTIONED** Item has options that must be specified for different features. The chosen options must match for two Lots of an **OPTIONED** Item (to be interchangeable). A **STANDARD** Item has no Configuration fields to be specified -- they are all considered identical (interchangeable). In contrast, a **CUSTOM** Item also has no Configuration-specific information, but they are all considered distinct (not interchangeable).

Note that currently, RHYTHM only implements Items with **STANDARD** configuration. Future releases will support more complex configuration related extensions.

Without this extra information, you may not know what you really have. The fields of this Item model provide the default values to be used by the Configuration.

Default: **STANDARD**
 Extensions:
STANDARD, CUSTOM,

lots_tracked - - *a Logical field of model Item*
 If "false", then the identity of Lots are not tracked. Once the lot reaches a Buffer, its quantity is lumped in with the rest.

If "true", then the identity of Lots are tracked. Buffers containing this Item will maintain a list of the Lots contained within. The purpose of this is to maintain lot-specific information, such as the Configuration, an expiration Date, concentrations, quality grades, test results, or source information. Such information may be used by downstream Operations in computing behavior.

Models	Item Model
--------	------------

Note that if the 'spec' extension requires additional fields, then 'lots_tracked' will be "true" (and unselectable to "false"). However, if the 'spec' does not require added fields, this may still be set to "true" in order to preserve user-defined or other lot-specific fields.

Default: true if 'spec' requires additional fields, false otherwise

delivery - - *a Operation field of model Item*
 The Operation or service performed to deliver this Item to the customer. This Operation must handle a **DELIVERY** motive. During planning, when operation plans are created for a specific operation, motive specifies reasons for creating those operation plans. For example, a transportation operation may be given a **TRANSIT** motive to transport an Item 1) from location A to location B. If the transportation operation can't handle **TRANSIT** motive, no operation plans can be planned to transport Item 1) using that operation. Similarly, delivery operation of this item must handle **DELIVERY** motive else, during planning no operation plans will be planned to deliver this item.

This Operation's primary role is to move the Item from a finished goods inventory Buffer to the delivery_address for that Delivery_Request.

A secondary role of this Operation is to choose which finished goods inventory Buffer to draw from if there is more than one that supplies this Item. Thus, this may be an **ALTERNATES**. * Operation that contains a sub_operation for each alternate Buffer. The **ALTERNATES**. * Operation may choose based upon the delivery_address of the Delivery_Request. For example, a Delivery_Request destined for Los Angeles may be supplied first from San Francisco, alternately from Dallas, and as a last resort from Atlanta.

Default: [unspecified]

buffers - - *a List(Buffer) field of model Item*
 All Buffers of the 'owner' Site that manage this Item.
 Properties: Export-Only Field

buffer_plans (Plan) - - *a List(Buffer_Plan) field of model Item*
 All Buffer_Plans in the specified Plan for the Buffers of the 'owner' Site that manage this Item. This buffer_plans(plan) is a shortcut for 'buffers_for_each(#buffer_plan(plan))'.
 Properties: Export-Only Field

Models	Item Model
--------	------------

unit -- a *Unit field of model Item*

The unit defines the Quantity of one unit of this Item, whether this Item is 'discrete' (can only exist in whole units), and what the 'preferred_measure' for this Item is. It can also 'convert' between different units of Measure and units of this Item.

For example, one unit of Item X may be "3 kg", "400 ml", and "\$12". It may be 'discrete', meaning that a need for "5 kg" will result in "6 kg" (2 units) being built. The 'preferred_measure' may be "kg" which will cause most fields that display a Quantity of this Item to display in "kg" by default.

Note that this field is not scalable -- you cannot assign it a new Unit. Rather, you can get the Unit of this Item and set that Unit's fields.

Properties: Export-Only Field

owner -- a *Site field of model Item*
The Site to which this Item belongs.
Properties: Export-Only Field

Models	spec submodels of model Item
--------	------------------------------

5.1.1.5 Buffer Model 5.1.1.4.1 spec submodels of model Item

Buffer -- a *submodel of model Site*

A Buffer models the management of the flow of interchangeable Items. Buffer handles most of the material/inventory planning functionality.

It can model all flow of a particular Item, or a subset of that. In modeling a supply chain, a Buffer typically only models the flow of Items at a particular Location (a SKU). Items at different Locations are usually not interchangeable (transportation is needed).

Smaller subsets can be defined. One Buffer could model Items at a Location to be supplied to a very important customer, while another models that Item supplied to other customers. The safety stock maintained for the special customer cannot be treated as interchangeable with the safety stocks for the lesser customers.

One Buffer could model Items at a Location built a certain way (by US Military-approved Resource), while another Buffer models the rest. Those are surely not interchangeable. In fact, it could be argued that such should be different Items. That is not required, but it is required that their flow is managed by different Buffers.

The Buffer defines interchangeability. All Items managed by a Buffer must be interchangeable.

Although most inventory planning functionality is handled by Buffer, it does not directly address the capacity needed to store the Items: it places Load on a Resource to model that needed storage capacity.

The Buffer provides logic to generate Operations / Demand as the result of the flow in and out of it. It manages safety stocks, safety times, lot sizing, and timing.

In each Site, there is a special unclonable Buffer named (unspecified). It has the default values for all fields. Since it has the UNMODELED_flow_policy extension, the flow from Operations that use the (unspecified) Buffer will not be modeled.

The model has selectors:
flow_policy, stocking_policy,

The Buffer model has these submodels :
Buffer_Problem_Detector.

Models	Buffer Model
--------	--------------

The Buffer model has fields that references these models :
Item, Location, Operation, Buffer_Problem_Detector, Unit, Buffer_Plan, Site.

These models have a field that is a Buffer model :

Buffer_Plan, LINK, Flow, Buffer_Problem_Detector, Flow_Criterion.

The key field for this model is name

This model may be extended with user-defined fields.

name - - *a Symbol field of model Buffer*

The name of this Buffer. This name must be unique among Buffers at this Site.

Default: None -- this is a key field

description - - *a String field of model Buffer*

A description of this Buffer.

Default: none

category - - *a Symbol field of model Buffer*

A simple user-defined categorization of Buffers. This is often used in reports. This will allow users to filter the Buffers by category, and to write expressions that depend upon the category of the Buffers.

The standard reports look for the values "Inventory", "Phantom", "Capacity", and "Cash". An "Inventory" Buffer contains physical Items, and can usually be further categorized by the Item category. Note that a "Resource" Item may be either in an "Inventory" Buffer or in a "Capacity" Buffer, depending upon the nature of it. A "Phantom" Buffer is in the model structure, but is best treated as if it was not. The Item.category is often "Phantom", but may not be (the Item may not be treated as a "Phantom" by all Buffers). A "Capacity" Buffer models capacity that is flow-limited rather than load-limited (e.g., subcontractor that commits to a certain quantity). The Item.category is usually "Resource". A "Cash" Buffer can be used to model cash flow. The Item.category is usually "Cash".

Default: Inventory

Item - - *a Item field of model Buffer*

The Item managed by this Buffer.

If this Item is not 'real' (e.g., [unspecified]), then this Buffer cannot be consumed from or produced into. Such a Buffer is typically only used as a 'family' for other Buffers.
Default: [unspecified]

Models	Buffer Model
--------	--------------

location - - *a Location field of model Buffer*

The Location of this Buffer. Note that this is not necessarily where the Items are stored -- that is defined by the storage_operation. Rather, this is the Location of the logical Flow point (staging area, etc.). Typically, though, the storage_operation's Location is the same as or within this Location.
Default: [unspecified]

flow_policy - - *an extension selector of model Buffer*

This extension defines how the Operation Flows placed on this Buffer are planned. The associated fields define the Problems that can be caused by the flow of Items through this Buffer, and how those Problems are resolved. In particular, it defines the relationship between consumers from the Buffer and producers into it. For example, Lot-For-Lot, Fixed, EOQ, and POQ inventory policies would be implemented as different flow_policy's.

Default: LFL_SIMPLE

Extensions:

BUCKETED_NESTED_SORT, PRODUCING_FLOW_CALENDAR,
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK,
SUPPLY_CALENDAR_ON_HAND_CALENDAR,
ON_HAND_CALENDAR_FILTER_AND_RANK,
FLOW_LIMIT_CALENDAR,
FLOW_LIMIT_CALENDAR_FILTER_AND_RANK, INFINITE_SUPPLIER,
BASIC_FILTER_AND_RANK, FIXED_QUANTITY_MULTIPLE,
MULTIPLE_FILTER_AND_RANK, FIXED_QUANTITY_FENCED,
FIXED_TIME, LFL_SIMPLE, LFL_BOUNDED, MLFL_BOUNDED,

stocking_policy - - *an extension selector of model Buffer*

Specifies formula and/or algorithm to calculate the safety stock (min_on_hand and/or min_time) using local and/or global information on supply, process and demand variability and customer service expectation. Note that the safety stock levels will be enforced by the buffer's flow_policy.

Note that setting stocking_policy to anything other than MANUAL is irrelevant for a non-inventory carrying flow_policy such as LFL_SIMPLE, SUPPLIER, etc.

Currently CALENDAR_stocking_policy is only implemented for the PRODUCING_FLOW_CALENDAR_flow_policy. CALENDAR_stocking_policy automatically computes time-phased safety stock. It also computes separate safety stock due to demand and supply variability. Note that the user can either bias or override the automatically computed safety stock by specifying a custom formula. See the CALENDAR_stocking_policy description for more details.

Models	Buffer Model
--------	--------------

Default: MANUAL
Extensions:
MANUAL, CALENDAR.

problem_detectors - - a list of Buffer_Problem_Detector submodels of model Buffer

A list of additional Problem Detectors that are applied to this Buffer. They are notified on each change in flow and have the opportunity to identify specific Problems.

Such Problems are resolved in a fairly disintegrated fashion compared to a 'flow_policy' designed to deal with the same issues, but the Problems are accurately identified. Some may require manual aid for intelligent resolution. Some will work well with certain 'flow_policy's, but not others.

The primary intent is to provide full flexibility and orthogonality in describing all of the Problems that need to be detected, even if an intelligent solver (in the form of a 'flow_policy' extension) has not been provided for that particular combination of Problems.

For example, there may be no 'flow_policy' that deals with perishing inventory, minimum safety stock, levels, and prioritized allocation decisions. Despite that, we may have individual Buffer_Problem_Detectors that can deal with each one of those, individually. By just dropping each of those into this list independently, all of those Problems will be accurately detected, supporting full visibility. However, there would be no algorithm designed for intelligently planning with that specific combination of four 'problem_detectors'.

level - - a Integer field of model Buffer

Every Buffer in a Site is assigned a level code signifying the relative level upstream from the end Item Buffers (the Buffers consumed only by delivery Operations). The end Item Buffers are level 0. The Buffers that are consumed by Operations that produce into end Item Buffers are level 1. And so on back through the 'producing_operation's of the Buffers. The value of this field is computed by traversing the FLO network of the owner site.

Level codes can be used algorithmically (as is done in traditional MRP) to traverse all of the Buffer_Plans in an orderly way such that all the requirements on a Buffer are computed prior to computing its requirements on other Buffers.

Level codes are also very useful in displaying information about the Buffers. By laying out the Buffers graphically according to the level codes, the flow of material can be kept in mostly one direction.

Models	Buffer Model
--------	--------------

Note that unlike the traditional definition of level codes, the 'level' values must accommodate cyclic material flow (for example, consider a mold that is consumed by an Operation, flows with the material in the routing until it is later removed, cleaned, and then returned to the Buffer from where it started). The 'level' values of Buffers in a cycle are computed from the Buffer with largest 'level' value that is produced from any of the Buffers in the cycle.

Buffers that are not consumed by the producing_operation of any other Buffer will be assigned level 0.

Properties: command=True Export-Only Field

all_producing_operations - - a List(Operation) field of model Buffer

A where-used analysis that returns all Operations that have a Flow that produces into this Buffer. Note that this is a Read-Only field that has no effect on the planning of this Buffer. (In contrast, the field 'producing_operation' is commonly used by 'flow_policy' extensions to specify which Operation should be planned to supply it when needed.) Properties: Export-Only Field

all_consuming_operations - - a List(Operation) field of model Buffer

A where-used analysis that returns all Operations that have a Flow that consumes from this Buffer. Note that this is a Read-Only field that has no effect on the planning of this Buffer. (In contrast, the field 'consuming_operation' is commonly used by 'flow_policy' extensions to specify which Operation should be planned to consume from it when needed.) Properties: Export-Only Field

unit - - a Unit field of model Buffer

The 'unit' defines the Quantity of one unit of this Buffer, whether this Buffer is discrete (can only hold items in whole units), and what the preferred_measure for this Buffer is. It can also 'convert' between different units of Measure and units of this Buffer.

For example, one unit of Buffer X may be "3 kg", "400 ml", and "\$12". It may be discrete, meaning that a need for "5 kg" will result in "6 kg" (2 units) being built. The 'preferred_measure' may be "kg" which will cause most fields that display a Quantity of this Buffer to display in "kg" by default.

Note that this field is not settable -- you cannot assign it a new Unit. Rather, you can get the Unit of this Buffer and set that Unit's fields.

Properties: Export-Only Field

Models	Buffer Model
--------	--------------

buffer_plan (Plan) -- *a Buffer_Plan field of model Buffer*
Returns the Buffer_Plan for this Buffer in the given Plan.
Properties: Export-Only Field

all_supplying_operations -- *a List(Operation) field of model Buffer*
Obsolete! This field has been renamed 'all_producing_operations' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.
Properties: obsolete=True Export-Only Field

owner -- *a Site field of model Buffer*
The Site to which this Buffer belongs.
Properties: Export-Only Field

Models	Buffer_Problem_Detector Model
--------	-------------------------------

5.1.5.1 Buffer_Problem_Detector Model

Buffer_Problem_Detector -- *a submodel of model Buffer*

A list of additional Problem Detectors that are applied to this Buffer. They are notified on each change in flow and have the opportunity to identify specific Problems.

Such Problems are resolved in a fairly disintegrated fashion compared to a flow_policy designed to deal with the same issues, but the Problems are accurately identified. Some may require manual aid for intelligent resolution. Some will work well with certain flow_policy's, but not others.

The primary intent is to provide full flexibility and orthogonality in describing all of the Problems that need to be detected, even if an intelligent solver (in the form of a flow_policy' extension) has not been provided for that particular combination of Problems.

For example, there may be no 'flow_policy' that deals with perishing inventory, minimum safety stock levels, and prioritized allocation decisions. Despite that, we may have individual Buffer_Problem_Detectors that can deal with each one of those, individually. By just dropping each of those into this list independently, all of those Problems will be accurately detected, supporting full visibility. However, there would be no algorithm designed for intelligently planning with that specific combination of four problem_detectors.

The model has selectors:
detector.

The Buffer_Problem_Detector model has fields that references these models :
Buffer.

These models have a field that is a Buffer_Problem_Detector model :
Buffer.

The key field for this model is detector
detector -- *an extension selector of model Buffer_Problem_Detector*
The problem to be identified, and possibly resolved.
Default: None -- this is a key field
Extensions:

Models	Buffer_Problem_Detector Model
--------	-------------------------------

feasible -- *a Logical field of model Buffer_Problem_Detector*
If "false", then this Problem represents an infeasibility that must be resolved to make the plan feasible. In that case, this will have infinite cost.

Setting this from "true" to "false" also sets cost to "oo". Setting this from "false" to "true" also sets cost from "oo" to "0". Note that this field is strictly unnecessary -- cost is sufficient -- this is provided purely for convenience.

Default: false
Properties: command=True

cost -- *a Money field of model Buffer_Problem_Detector*

The cost incurred if this Problem is not resolved. If infinite, then this Problem represents an infeasibility that must be resolved to make the plan feasible. In that case, the field 'feasible' will be "false".

Setting this to "oo" also sets 'feasible' to "false". Setting this to a finite value also sets 'feasible' to "true".
Default: oo

owner -- *a Buffer field of model Buffer_Problem_Detector*

Properties: Export-Only Field

Models	flow_policy submodule of model Buffer
--------	---------------------------------------

5.1.1.5.2 flow_policy submodels of model Buffer

5.1.1.5.3 Flow_Criterion Model

Flow_Criterion -- *a submodel of model Buffer*

The criterion and relative rank for judging the rank of the Flow_Plan. It generally corresponds to one of the many properties of the Flow_Plan, its Operation_Plan, or the Delivery_Request or Delivery_Promise that motivated it.

The model has selectors:
criterion.

The Flow_Criterion model has fields that references these models :
Buffer:

These models have a field that is a Flow_Criterion model :
BUCKETED_NESTED_SORT, BUCKETED_COMBINED_SORT.

The key field for this model is criterion

criterion -- *an extension selector of model Flow_Criterion*

The criterion for judging the rank of the Flow_Plan. It generally corresponds to one of the many properties of the Flow_Plan, its Operation_Plan, or the Delivery_Request or Delivery_Promise that motivated it.

Default: None -- this is a key field

Extensions:

CUSTOMER_RANK, SELLER_RANK, REQUEST_RANK,
ACTUAL_OR_FORECAST, REQUEST_ISSUED, PROMISE_DUE,
REQUEST_DUE, DUE_DATE, PROMISED, ENTRY_DATE,

rank -- *a Number field of model Flow_Criterion*

The importance, weight, or rank of this criterion relative to the others. Some policies may sort by it, others may mathematically weigh by it.

Note, as with all rank's, the higher the Number, the higher the rank, the greater the importance, the larger the weight.
Default: 0

owner -- *a Buffer field of model Flow_Criterion*

Properties: Export-Only Field

<i>Models</i>	<i>Resource Model</i>
---------------	-----------------------

5.1.1.6 Resource Model

Resource -- *a submodel of model Site*

A Resource models the capacity to perform Operations. This includes machines, tools, fixtures, labor, trucks, molds, dies, masks, and other things that are used by Operations in causing Flow between Buffers. It also includes storage space, containers, racks, and other things that are used to hold Items within Buffers or during Operations.

Fundamentally, a Resource provides a certain efficiency level that may change over time, which defines its basic capacity. It also has a variability level that defines how to compensate the plans to prevent the variability from invalidating them. It may have setup Operations imposed between normal Operations, and maintenance Operations at fixed or computed dates.

The intelligent automated planning of the Loads on the Resource is done as specified by the Resource's load_policy, which can impose various rules, restrictions, and conditions on the loading of the Resource. In addition, for maximum user flexibility, there is an extra list of Resource_Problem_Detectors that can each impose certain restrictions or conditions on the Resource.

In each Site, there is a special uneditable Resource named [unspecified]. It is the root 'family' to which all Resources of the 'owner' Site belong, directly or indirectly. It has the default values for all fields. Since it has the INFINITE load_policy extension, the load from Operations that use the [unspecified] Resource is ignored.

The model has selectors:

efficiency, variability, maintenance, size, load_policy,

The Resource model has these submodels :

Resource_Skill, Resource_Problem_Detector.

The Resource model has fields that references these models :

Location, Resource_Skill, Operation, Resource_Problem_Detector, Resource_Plan, Site.

These models have a field that is a Resource model :

Resource_Plan, LINK, Load, Resource_Skill, Resource_Problem_Detector, Skill_Resource, MPPS_Operation_Resource, MPPS_Batch, PRIMARY, PREFER_PRIMARY, Size_Dimension.

<i>Models</i>	<i>Resource Model</i>
---------------	-----------------------

The key field for this model is name
This model may be extended with user-defined fields.

name -- *a Symbol field of model Resource*

The name of this Resource. This 'name' must be unique among Resources at this Site.

Default: None -- this is a key field

description -- *a String field of model Resource*

A description of this Resource.

Default: none

category -- *a Symbol field of model Resource*

A simple user-defined categorization of Resources. This is often used in reports. This will allow users to filter the Resources by category, and to write expressions that depend upon the category of the Resources.

The standard reports look for the values "Labor", "Manufacturing", "Transportation", "Tool", "Fixture", "Rework", "Setup", and "Maintenance".

Default: Machine

sub_category -- *a Symbol field of model Resource*

A simple user-defined categorization of Resources. This is often used in reports. This will allow users to filter the Resources by category, and to write expressions that depend upon the category of the Resources.

The standard reports do not look for any specific values. It is commonly used to categorize by mode, method, duration, cost, or usage pattern. For example, in Transportation, this may be "Express", "Air", "Rail", "Truck", or "Vessel". But any values can be used -- whatever is useful for categorization in reports.

Default: [unspecified]

location -- *a Location field of model Resource*

The Location of this Resource, if restricted or fixed. Note that even a movable

Resource may specify a Location here that restricts the Resource to Locations within it. For example, a forklift may be able to move to any of 12 docks in Building One, but cannot move to the docks in Building Two. Thus, its 'location' may be 'Building One'. If [unspecified], then the Resource is allowed to move anywhere.

Default: [unspecified]

Skills - - a list of Resource_Skill submodels of model Resource

The Skills of which this Resource is capable, and its efficiency at those Skills. Note that if the Resource's efficiency at a certain Skill is specified 80%, but the Resource is planned to be operating at 50% efficiency on a Date, then its planned efficiency at that Skill on that Date is only 40% (50% of 80%).

efficiency - - an extension selector of model Resource

Defines how the availability, capacity, and efficiency of this Resource is modeled. The associated fields describe the efficiency of this Resource over time, as well as policies for changing/increasing the efficiency.

The efficiency indicates how well the Resource is performing. It affects the time required by Operations: the standard time (std_time) is based on 100% efficiency. The actual time (time) required for an Operation is the standard time divided by the current efficiency level.

Note that efficiency greater than 100% is legal and supported. For example, a newer machine may operate much faster than the older machines; rather than changing all the Operation times, the new machine can be modeled as having higher than standard (100%) efficiency.

A 0% efficiency effectively indicates unavailability. Thus, the efficiency fields define availability: any time with 0% efficiency is unavailable time; the remainder is the available.

As traditionally defined, capacity is the available hours multiplied by the efficiency during those hours. Thus, the efficiency fields fully define the capacity of this Resource. Note that the capacity of a Resource during a Date_Range is the amount of load in standard time that can be handled; the available time of a Resource during a Date_Range is the amount of load in actual time that can be handled.

For example, if a Resource has an efficiency of 50% from 8am to 6pm each day, and 0% the rest of the day, then it is available 10 hours per day, and has a capacity of 5 standard hours each day. If an Operation that takes 3 standard hours is loaded on the Resource, then it will take 6 hours of time (3hr / 50%). The utilization of the Resource can be computed as either 3 / 5 in standard hours, or 6 / 10 in actual hours (60% either way).

Machines typically have continuous availability. Operators, human resources, typically work shifts. However, if operators are not explicitly modeled, a work center may be modeled as working shifts, to reflect the availability of operators to run it.

Efficiency may be used to reflect regular downtime. For instance, if a machine typically goes down for 6 minutes out of every hour, a convenient way to account for that lost capacity would be to reduce efficiency by 10% (6/60), to 90%. For modelling less regular downtime, see the 'Variability' extension.

Efficiency may drop with total load time, until a maintenance Operation is performed to restore it to full performance. Or, efficiency may gradually ramp up over time to reflect a learning curve as a new Resource is being mastered. Or, efficiency may be adjusted in a shift calendar to reflect lower efficiency on night or weekend shifts. And so on.

Default: FIXED, which defaults to 100% efficiency all day, every day

Extensions:

FIXED, CALENDAR,

variability - - an extension selector of model Resource

Defines how the expected variability of this Resource is modeled and compensated. A common source of variability is irregular, but not uncommon, failures. Thus, the associated fields may describe frequency of failure (e.g., MTBF -- mean time between failures) and the expected downtime due to each failure (e.g., MTRR -- mean time to repair).

There are many other sources of variability as well. Variability of efficiency or availability is quite common. If the efficiency can vary from 20% to 90%, but it averages 85%, then how should it be modeled?

Modelling at 85% efficiency is not quite satisfactory -- there will often be times that this is not achieved. Thus, plans based on that will often be infeasible. Modelling at other than 85% (the average) will definitely, over the long term, either under- or over-estimate the available capacity.

The right answer is to model at 85% efficiency, but compensate for the variability by buffering between the Operations on this Resource and downstream Operations by enough time to cover under-performance. In that way, the overall efficiency is modeled properly, at the average, but the plan is buffered such that under-performance does not invalidate it.

Thus, the primary job of the fields associated with the 'Variability' extension is to define the time padding added before and after the Operations on this Resource to properly compensate for the variability (prevent the variability from adversely affecting the feasibility of the plans).

An increase in padding is accompanied by an equivalent increase in WIP for routings that utilize this Resource. Thus, the pads should generally be kept as small as possible without making the plan too brittle in the face of disturbances.

Note that the WIP created by these pads may not sit around this Resource. If on the floor, the next Operation on the dispatch list is always performed as soon as it is available, then the WIP will naturally tend to migrate to in front of the constraint Resource(s) downstream from this Resource. Note that this is the desired behavior: the primary purpose of this padding is to protect the constraint Resource(s) from becoming under-utilized due to the variability of this Resource.

Default: ZERO

Extensions:

ZERO, FIXED,

maintenance - - *an extension selector of model Resource*

Defines how maintenance is specified for this Resource. The associated fields describe when maintenance should be performed and what Operation is used to perform it. It may describe both major and minor maintenance Operations. It may base timing either total time, loaded time, setup time, regular calendar intervals, or many other criteria. The Operations may use particular other Resources, which may be available only at certain dates.

Default: ZERO

Extensions:

ZERO,

size - - *an extension selector of model Resource*

Defines the size limits on the loads that can be placed on this Resource. The associated fields describe the size (volume, weight) limits of this Resource over time. The size fields indicate how much Load can be handled at once. For example, if the size limit is "2 tons", then up to 2 tons of Loads may be simultaneously processed.

Note that a unitless Quantity as a size typically indicates the ability to process that number of Loads at a time (effectively N identical resources). Thus, a very common size is simply "1". (The default size is the extension "FIXED" which defaults to "1".)

Note that the ability to place multiple loads on a Resource at the same time is defined by the 'load_policy'. For example, SHARED_USE and SIMULTANEOUS_USE allow multiple loads; whereas, EXCLUSIVE_USE does not. Given a SHARED_USE load_policy Resource, the size fields limit how much can be loaded at once.

Default: UNLIMITED

Extensions:

UNLIMITED, FIXED_COUNT, FIXED_QUANTITY, CALENDAR_COUNT, MULTI_DIMENSION,

load_policy - - *an extension selector of model Resource*

Defines how the Operation Loads placed on this Resource are planned. The associated fields describe the rules and restrictions on the Loads and the Problems that could occur due to usage of this Resource's capacity. Although many of the Resource characteristics are modeled independent of this extension, any intelligent algorithms to deal with those characteristics will be specified here.

For example, the size limits of this Resource are specified by the size extension and related fields; however, the constraints on shared usage of that size and how to intelligently plan the Operation Loads in order to maximize the use of that size are specified by the 'load_policy' and related fields.

Thus, the load_policy extensions selected may depend upon the other extensions selected.

Default: INFINITE

Extensions:

SIMPLE_CONSOLIDATION, INFINITE_USE, EXCLUSIVE_USE, SHARED_USE,

problem_detectors - - *a list of Resource_Problem_Detector submodels of model Resource*

Do not use Resource_Problem_Detectors if there is a 'load_policy' extension that can be used to model the Resource.

This is a list of additional Problem Detectors that are applied to this Resource. They are notified on each change in load or capacity and have the opportunity to identify specific Problems.

Such Problems are resolved in a fairly disintegrated fashion compared to a 'load_policy' designed to deal with the same issues, but the Problems are accurately identified. Some may require manual aid for intelligent resolution. Some will work well with certain load_policy's, but not others.

The primary intent is to provide full flexibility and orthogonality in describing all of the Problems that need to be detected, even if an intelligent solver (in the form of a 'load_policy' extension) has not been provided for that particular combination of Problems.

For example, there may be no 'load_policy' that deals with sequence-dependent setup, maintains a minimum utilization, disallows "green" jobs on Mondays, and prevents four related Resources from all being loaded at once. Despite that, we may have individual Resource_Problem_Detectors that can deal with each one of those, individually. By just dropping each of those into this list independently, all of those Problems will be accurately detected, supporting full visibility. However, there would be no algorithm designed for intelligently planning with that specific combination of four problem_detectors:

```
resource_plan (Plan) -- a Resource_Plan field of model Resource
Returns the Resource_Plan for this Resource in the given Plan.
Properties:    Export-Only Field

owner -- a Site field of model Resource
The Site to which this Resource belongs.
Properties:    Export-Only Field
```

5.1.1.6.1 Resource_Skill Model

Resource_Skill -- a submodel of model Resource

The Skills of which this Resource is capable, and its efficiency at those Skills. Note that if the Resource's efficiency at a certain Skill is specified 80%, but the Resource is planned to be operating at 50% efficiency on a Date, then its planned efficiency at that Skill on that Date is only 40% (50% of 80%).

The model has selectors:
efficiency.

The Resource_Skill model has fields that references these models:
Skill, Resource.

These models have a field that is a Resource_Skill model:
Resource.

The key field for this model is skill

skill -- a Skill field of model Resource_Skill
A skill of which this Resource is capable.
Default: None -- this is a key field

efficiency_level (Date) -- a Percentage field of model Resource_Skill
The efficiency of the 'owner' Resource in performing the 'skill' on the given Date. The efficiency may vary with Date, as specified by the 'efficiency' extension.
Properties: Export-Only Field

efficiency -- an extension selector of model Resource_Skill
Specifies the efficiency of the 'owner' Resource in performing 'skill'. Various curves are possible, supporting learning curves, ramp-up, and any other variation with Date. The efficiency curve can be FIXED, or from a CALENDAR.
Default: FIXED
Extensions:
FIXED, CALENDAR.

owner -- a Resource field of model Resource_Skill
The Resource that is capable of this 'skill' with this 'efficiency'.
Properties: Export-Only Field

Models	efficiency submodels of model Resource
--------	--

5.1.1.6.2 efficiency submodels of model Resource

Models	size submodels of model Resource
--------	----------------------------------

5.1.1.6.3 size submodels of model Resource

5.1.1.6.4 Size_Dimension Model

Size_Dimension -- a submodel of model Resource

The various dimensions of the size of each one of the aggregate resource. The dimensions need not be orthogonal.

The Size_Dimension model has fields that references these models : Resource.

These models have a field that is a Size_Dimension model : MULTI_DIMENSION.

The key field for this model is name

name -- a Symbol field of model Size_Dimension
The name of this dimension of the size
Default: None -- this is a key field

min_size -- a Quantity field of model Size_Dimension
The minimum total size of Operation_Plans that can run on this Resource at once.
Default is 0.
Default: 0

max_size -- a Quantity field of model Size_Dimension
The maximum total size of Operation_Plans that can run on this Resource at once.
Default is infinite.
Default: oo (infinite, unlimited, unless)

owner -- a Resource field of model Size_Dimension

Properties: Export-Only Field

Models	load_policy submodels of model Resource
--------	---

5.1.1.6.5 load_policy submodels of model Resource

5.1.1.6.6 Resource_Setup_Order Model

Resource_Setup_Order -- a submodel of model Resource

These models have a field that is a Resource_Setup_Order model :

SHARED_USE_SDS, SIMULTANEOUS_USE_SDS.

5.1.1.6.7 Resource_Blocks Model

Resource_Blocks -- a submodel of model Resource

These models have a field that is a Resource_Blocks model :

BLOCK_CALENDAR, BLOCK_CYCLE.

5.1.1.7 Skill Model

Skill -- a submodel of model Site

A Skill models a basic capability needed by a resource to perform an Operation. Different Resources may be capable of the same Skill, but possibly at different efficiencies. An Operation can specify a Skill, then during the planning, any Resource capable of that Skill can be used. All the resources within one skill group are considered alternatives of each other.

Note that the (unspecified) Resource results in a similarly special, unmediable Skill named (unspecified) which contains that one (unspecified) Resource with efficiency "100%".

The model has selectors:

selection.

The Skill model has these submodels :

Skill_Resource.

The Skill model has fields that references these models :

Skill_Resource, Site.

These models have a field that is a Skill model :

Resource_Plan, LINK, Load, SAME_RESOURCE, Resource_Skill, Skill_Resource.

Models	Skill Model
--------	-------------

The key field for this model is name

This model may be extended with user-defined fields.

name -- a Symbol field of model Skill

The name of this Skill. This 'name' must be unique among Skills at this Site.

Default: None -- this is a key field

description -- a String field of model Skill

A description of this Skill.

Default: none

resources -- a list of Skill. Resource submodels of model Skill

The Resources that have this Skill, and their efficiencies at this Skill.

selection -- an extension selector of model Skill

Describes the logic to be used to select one of the resources from all the alternate resources within this skill group. For example, MAX_EFFICIENCY selects a resource among alternates based on the highest efficiency.

Default: MAX_EFFICIENCY

Extensions:

PRIMARY, PREFER, PRIMARY, EVEN, MAX_EFFICIENCY,

loading_operations -- a List(Operation) field of model Skill

A where-used analysis that returns all Operations that need this Skill.

Properties: Export-Only Field

loading_buffers -- a List(Buffer) field of model Skill

A where-used analysis that returns all Buffers that has supplying, receiving, storage or picking operation that needs this Skill.

Properties: Export-Only Field

owner -- a Site field of model Skill

Properties: Export-Only Field

5.1.1.7.1 Skill_Resource Model

Skill_Resource -- a submodel of model Skill

The Resources that have this Skill, and their efficiencies at this Skill.

The model has selectors:
efficiency.

The Skill_Resource model has fields that references these models :
Resource, Skill.

These models have a field that is a Skill_Resource model :
Skill.

The key field for this model is resource

resource -- a Resource field of model Skill_Resource

A Resource capable of the 'owner' Skill.

Default: None -- this is a key field

efficiency_level (Date) -- a Percentage field of model Skill_Resource

The efficiency of 'resource' in performing 'owner' Skill on the given Date. The efficiency may vary with Date, as specified by the 'efficiency'.

Properties: Export-Only Field

efficiency_profile (Date_Range) -- a List(Profile_Percentage) field of model

Skill_Resource

A List of all changes in the efficiency profile during the specified finite Date_Range.

This efficiency changes are specific to 'resource' while performing the owner skill.

Properties: Export-Only Field

efficiency -- an extension selector of model Skill_Resource

Specifies the efficiency of 'resource' in performing the 'owner' Skill. Various curves are possible, supporting learning curves, ramp-up, and any other variation with Date. The efficiency can be FIXED or from a CALENDAR.

Default: FIXED

Extensions:

FIXED, CALENDAR.

owner -- a Skill field of model Skill_Resource

The Skill that this 'resource' is capable with this 'efficiency'.

Properties: Export-Only Field

5.1.1.8 Operation Model

Operation -- a submodel of model Site

Operation Definition

Operation is a submodel of the Site model. An Operation models a process, activity, or action that transforms or moves items. This results in a flow into, out of, or between buffers.

Operations may required Resources with specific Skills, modeled by Loads. Those Resources model the capacity to perform Operations. Flows model the flow of items to and from Buffers that result from Operations.

An Operation has a "unit" of its own, which can be specified in one or more Measures. Thus, one unit of an Operation may be 5 liters, 8.78 kg, and \$36.50.

All proportional Times and Quantities associated with an Operation are relative to that unit. The proportional Quantities generally have the suffix _per meaning per unit of the Operation.

Operations can be defined to be discrete. This means they cannot run in fractional units.

Example

If the Flows define that 8 legs and 2 tops produce 2 tables, then one unit of the Operation produces 2 tables. The times that are given in the process' extension will be for one unit, or 2 tables. So if the run_time is 1 hour per unit, then it takes 1 hour to produce 2 tables.

Operation Model Behavior

An Operation can use any number of Resources, with different run_times, and with staggered start and end dates. Therefore, a single activity or a whole series of activities can be modeled with a single Operation. This is important for having sufficient flexibility to model the diverse activities in different manufacturers, even within a

single supply chain. It is also important for the planner/modeler to understand, so that it can be used to maximum advantage when deciding what to model and how to model it.

In particular, note that an Operation can model a complete routing. A routing transforms or moves Items and it uses Resources to do it, which is the definition of an Operation. Further, with the extensible model architecture, the behavior of an Operation can be defined in many different ways.

In particular, an Operation can be defined in terms of other Operations (e.g. a routing). An Operation can consist of other Operations that are run in sequence. The relationships between those sub_operations can be different depending upon the chosen extension. So, an Operation can model a simple routing (a sequence of Operations allowed to spread), a flow routing (a sequence of Operations which must flow into one another), a set of Operations that can be run at the same time, alternates (a set of alternate Operations), and other combinations (see the process extension for examples).

An Operation can also model simply the bill-of-materials for an Item (Operations that just model the transform of Items, but do not model the capacity required to do that). Similarly, an Operation can model simply the bill-of-distribution for an Item (Operations that just model the movement of Items between Buffers, but do not model the capacity required to do that). Other Operations can then combine the bill-of-materials or bill-of-distribution Operations with Operations that properly consume capacity. Such separation is common in older manufacturing databases.

Note that some Operation specifications could define multiple Operations. For example, some manufacturing software defines Operations such that Resources can be loaded during portions of the Operation time. For ease of interfacing with such packages or databases, process extensions can be provided with the identical specification.

Example

You may have a simple routing Operation that consists of five Operations. One of those five may be an alternate Operation, which consists of four possible Operations.

One of those may itself be a simple routing Operation consisting of three Operations. Therefore, depending upon which alternate is chosen, you may have five Operations or seven Operations to perform.

Advantage of Operation Model Design

The advantage of the Operation model design is simplicity and consistency. By providing a simple building block and the flexibility to combine those blocks, a great deal of modeling capability is provided without complicating the common, simple cases. In this way, the critical Operations and Resources can be modeled in adequate detail while the non-critical models remain suitably inexpensive.

Submodels

The Operation Model consists of two submodels that are essential to its definition: the Flow and Load submodels. Complete understanding of the Operation Model cannot be achieved without understanding those submodels.

Submodel: Flow

The Flow model defines the consumption of Items from a Buffer or the production of Items to a Buffer. The Flow model contains the bill-of-material information. Each Flow can define an independent usage_policy that defines how much is consumed or produced per unit of the Operation and any yield information. The Flow can also define transfer batches which cause the material to flow at the rate of the Operation (rather than in one lump). Generally, consuming Flow-s occur at the beginning of the Operation: the first transfer batch flows at the start of the Operation and the rest follows at the rate of the Operation until the full Flow has been transferred. Similarly, producing Flow-s generally occur at the end of the Operation: a full transfer batch will occur at the end of the Operation and the rest will be transferred at the rate of the Operation preceding the end such that the full Flow is transferred. That pattern is common, but process extensions can define alternative Flow patterns. In particular, many fixed-time process-es do not have a rate, and thus transfer the full consuming Flow at the start and the full producing Flow at the end.

Models	Operation Model
--------	-----------------

See the Flow model for further detail.

Submodel: Load

The Load model defines the capacity required to perform the Operation. For each Load model in the Operation model, one or more Resources will be loaded by that Operation simultaneously. Each Load can specify a single Resource to be loaded, or a Skill group made up of numerous Resources. Generally each Load is applied for the full duration of the Operation. However, that can vary with each process extension. Any process extension that applies the Loads differently will describe that in its documentation. See the Load model for further detail.

Operation Model Default Value

In each Site, there is a special uneditable Operation named [unspecified]. It is the root 'family' to which all Operations of the tower Site belong, directly or indirectly. It has the default values for all fields. Since it has no Loads, no Flows, and takes no Time, it is suitable as a do-nothing Operation for the default value of most Operation fields. For example, the default value of the Resource 'setup_operation' is [unspecified]. The name [do_nothing] might be more accurate, but would be inconsistent with the many other [unspecified] models.

Operation Model Selectors

The Operation Model has the following selectors:

process

Operation Model Submodels

The Operation Model has the following submodels:

Flow

Load

Operation_Problem_Detector

Operation Model Fields

Models	Operation Model
--------	-----------------

The Operation Model has fields that reference these models:

Flow Load
Operation_Problem_Detector Site Unit

Models with Operation Model Field

These models have a field that is an Operation Model:

ACCESSORY Alternate_Operation BASIC
BUCKETED_COMBINED_SORT BUCKETED_NESTED_SORT Buffer CALEN-
DAR DUAL_WEIGHTED_TIME EARLIEST
Effective_Calendar_Operation FIXED_QUANTITY FIXED_QUANTITY_FENCED
FIXED_TIME Flow FLOW_MIN_ON_HAND
FLOW_MIN_TIME Flow LFL_MIN_TIME LFL_SIMPLE LINK
LFL_BOUNDED LFL_MIN_TIME MULT_BOUNDED
Load MULTIPLE LOAD_TIME
MPPS_Operation_Resource MULTIPLE Operation_Model_Detector
Operation_Plan OPTION PHANTOM
PRODUCING_FLOW_CALENDAR Request_Alternates Resource
REWORK Routing_Operation SAME_RESOURCE
Unordered_Operation WRAPPER

Operation Model Key Field

The model has selectors:

process,

The Operation model has these submodels :

Load, Flow, Operation_Problem_Detector.

The Operation model has fields that references these models :

Load, Flow, Operation_Problem_Detector, Unit, Site.

These models have a field that is a Operation model :

Operation_Plan, Resource, Buffer, LINK, Item, Load, Flow,
Operation_Problem_Detector, SAME_RESOURCE, Alternate_Operation,
Effective_Calendar_Operation, ACCESSORY, OPTION, EARLIEST,
MPPS_Operation_Resource, Request_Alternates, Routing_Operation,
Unordered_Operation, WRAPPER, REWORK, LOAD_TIME,
DUAL_WEIGHTED_TIME, CALENDAR, BUCKETED_NESTED_SORT,

Module	Operation Model
--------	-----------------

consume_flows - - *a List(Flow) field of model Operation*
consume_flows Field The Flows that are being consumed from Buffers by this Operation. To edit the Flows on this Operation, see Flows'.

Note that this List does not include Flows applied by any 'sub_operations'. See **all_consume_flows** for a List that includes the Flows of the sub_operations'.
 Properties: Export-Only Field

produce_flows - - *a List(Flow) field of model Operation*
produce_flows Field The Flows that are being produced to Buffers by this Operation. To edit the Flows on this Operation, see Flows'.

Note that this List does not include Flows applied by any 'sub_operations'. See **all_produce_flows** for a List that includes the Flows of the sub_operations'.
 Properties: Export-Only Field

all_consume_flows - - *a List(Flow) field of model Operation*
all_consume_flows Field The Flows that are being consumed from Buffers by this Operation or any of its sub_operations'. Note that the flows for all alternate Operations will show up, not just the "selected" ones. The "selected" one only has meaning for an Operation_Plan.
 Properties: Export-Only Field

all_produce_flows - - *a List(Flow) field of model Operation*
all_produce_flows Field The Flows that are being produced to Buffers by this Operation or any of its sub_operations'. Note that the flows for all alternate Operations will show up, not just the "selected" ones. The "selected" one only has meaning for an Operation_Plan.
 Properties: Export-Only Field

all_flows - - *a List(Flow) field of model Operation*
all_flows Field The Flows that are being produced to and consumed from Buffers by this Operation or any of its sub_operations'. Note that the flows for all alternate Operations will show up, not just the "selected" ones. The "selected" one only has meaning for an Operation_Plan.
 Properties: Export-Only Field

sub_operations - - *a List(Operation) field of model Operation*
sub_operations Field Operations that will be performed as part of this Operation. Every Operation in this List will have this Operation in its super_operations' List.

Module	Operation Model
--------	-----------------

For example, a ROUTING Operation (see the 'process' extension) will specify a sequence of other Operations that will be performed. Those Operations will appear in this list for analysis purposes. Note that the Operations are not specified here -- they are specified in the fields of the appropriate extension.

Properties: Export-Only Field

super_operations - - *a List(Operation) field of model Operation*
super_operations Field Operations that specify this Operation as a sub_operation. Every Operation in this List will have this Operation in its sub_operations' List.

For example, if this Operation is a step of a ROUTING Operation (see the 'process' extension), then that ROUTING Operation will appear in this list. Similarly, if this Operation is one of the alternates of an ALTERNATES Operation, then that ALTERNATES Operation will appear in this list.
 Properties: Export-Only Field

process - - *an extension selector of model Operation*
process Field The description of the process performed by this Operation, restrictions on that process, and rules on how to resolve Problems.

For example, a BASIC Operation (the default) models a time delay that depends upon the number of units of the Operation being performed. The DELAY_ONLY_FIXED Operation just models a fixed time delay no matter how many units are involved.

Different processes can define the behavior of the Operation in terms of other Operations. For example, there are various ALTERNATES process extensions which are defined to behave like one of several other Operations. There are various ROUTING process extensions which are defined to perform several other Operations in sequence. Other processes will perform other Operations in an unspecified ordering.
 Default: BASIC

Extensions:
 ALTERNATES_PRIMARY, ALTERNATES_PROPORTIONAL,
 EFFECTIVE_CALENDAR_DELAY_ONLY_FIXED, DELAY_ONLY_BASIC,
 BASIC_CALENDARS_FIXED_TIME, TIME_MULTIPLE_BASIC,
 BASIC_DELAYED_REQUEST_FIXED,
 REQUEST_FIXED_WITH_ANALYSIS, ROUTING.

problem_detectors - - *a list of Operation_Problem_Detector submodels of model Operation*
problem_detectors.Field A list of additional Problem Detectors that are applied to this Operation. They are notified on each change to quantity or dates of an Operation_Plan such that they can identify specific Problems.

Such Problems are resolved in a fairly disintegrated fashion compared to a 'process' designed to deal with the same issues, but the Problems are accurately identified. Some may require manual aid to be resolved well. Some will work well automated with certain 'process's, but not with others.

The primary intent is to provide full flexibility and orthogonality in describing all of the Problems that need to be detected, even if an intelligent solver (in the form of a 'process' extension) has not been provided for that particular combination of Problems.

For example, there may be no 'process' that Despite that, we may have individual Operation_Problem_Detectors that can deal with each one of those, individually. By just dropping each of those into this list independently, all of those Problems will be accurately detected, supporting full 'visibility'. However, there would be no algorithm designed for intelligently planning with that specific combination of four 'problem_detectors'.

planning_buffers - - *a List(Buffer) field of model Operation*

planning_buffers.Field A where-used analysis that returns all Buffers that will generate a plan for this Operation (e.g., to adjust their flow or store their items).

Properties: Export-Only Field

planning_resources - - *a List(Resource) field of model Operation*

planning_resources.Field A where-used analysis that returns all Resources that will generate a plan for this Operation (e.g., to setup, maintain, or replenish the Resource).

Properties: Export-Only Field

operation_plans (Plan) - - *a List(Operation_Plan) field of model Operation*

operation_plans (Plan).Field Returns a List of the Operation_Plans of this Operation planned in the specified Plan.

Properties: Export-Only Field

unit - - *a Unit field of model Operation*

unit.Field The unit defines the Quantity of one unit of this Operation, whether this Operation is discrete (can only exist in whole units), and what the preferred 'measure' for this item is. It can also 'convert' between different units of Measure and units of this Operation.

For example, one unit of Operation X may be "3 kg", "400 ml", and "\$12". It may be discrete, meaning that a need for "5 kg" will result in "6 kg" (2 units) being built. The preferred 'measure' may be "kg" which will cause most fields that display a Quantity of this item to display in "kg" by default.

Note that this field is not settable -- you cannot assign it a new Unit. Rather, you can get the Unit of this Operation and set that Unit's fields.

Default: 0

Properties: Export-Only Field

release_fence - - *a Horizon, Date field of model Operation*

release_fence.Field An offset from the "current" date of the plan, by which operation plans should have already been released. Top operation plans with a scheduled start which are not released by the release_fence will cause a UNRELEASED problem to be raised.

Note that problem detection is performed based solely on the release_fence of the top operation plan of an operation plan tree; settings for sub_operation plans are not considered.

Default: 0

release_fence_date (Plan) - - *a Date field of model Operation*

release_fence_date (Plan).Field Given a Plan, which return the date the Operations release_fence becomes effective.

Default: 0

Properties: Export-Only Field

release_soon_fence - - *a Horizon, Date field of model Operation*

release_soon_fence.Field An offset from the "release_fence" of the operation plan, which serves as warning that the operation plan should be released. Top operation plans with a scheduled start earlier than the release_soon_fence will cause a NEEDS_RELEASE problem to be raised.

Note that problem detection is performed based solely on the release_soon_fence of the top operation plan of an operation plan tree; settings for sub_operation plans are not considered.

Default: 0

release_soon_fence_date (Plan) -- *a Date field of model Operation*
release_soon_fence_date Field Given a Plan, which return the date the Operations
release_soon_fence becomes effective.

Default: 0
Properties: Export-Only Field

release_name_expression -- *a Expression field of model Operation*
release_name_expression Field An expression used to calculate the release name to be
used for operation plans based on this operation when they are released without specifying
a release name. Note that '#' in the expression refers to the Operation_Plan, not
the Operation.

Default: #operation.name & "-" & #operation.next_release_number:string
Properties: command=True

next_release_number -- *a Integer field of model Operation*
next_release_number Field When operation plans are released without specifying a
release name, a release name is generated for them, based on an expression (see the
release_name_expression field). This expression can contain an operation-specific,
unique integer so that the release names for all operation plans corresponding to a particular
operation are different. **next_release_number** provides the next number in the
sequence and is intended for use in **release_name_expression**. Using
next_release_number in an expression will cause the internal counter to increment
each time it is evaluated.

Default: 1
Properties: command=True Export-Only Field

release_number -- *a Integer field of model Operation*
release_number Field The last release number generated by **next_release_number**.
When the value is zero, no release numbers have yet been generated. The value may
be set, in which case the next release number generated with **next_release_number** will
be 1 greater than the value set. Note that setting this value smaller than the largest
value already used may cause generated names to be duplicated.

owner -- *a Site field of model Operation*
owner Field The Site to which this Operation belongs.
Properties: Export-Only Field

5.1.1.8.1 Load Model

Load -- *a submodel of model Operation*

The Load model defines the loads put on Resources by the 'owner' Operation. Each
Load specifies either a 'resource' or a 'skill' needed to perform the 'owner' Operation,
the 'start_location' needed to prepare the Resource for the 'owner' Operation, and the
'start_location' of the 'owner' Operation. It also specifies the 'end_setup' and
'end_location' of the Resource following the 'owner' Operation.

Generally, all of the Loads of an Operation are applied simultaneously from the start of
the Operation to the end. However, different Operation 'process' extensions can specify
that they apply the Loads differently.

The 'usage_policy' extension of Load specifies how Resources are selected to be
loaded. For example, a RESOURCE 'usage_policy' Load simply uses the one
'resource'; a ONE 'usage_policy' Load uses one of the Resources capable of the 'skill';
a MAX 'usage_policy' Load simultaneously uses one or more of the Resources capable
of the 'skill' up to a maximum number. In the latter case, how the efficiency, and thus
the rate of the Operation, increases with higher numbers of Resources is specified by
the fields of the 'usage_policy'.

Note that even for a ONE Load, a Load that specifies a 'skill' may use multiple
Resources during the course of an Operation, just not simultaneously. Consider an
Operation which needs a Saw and someone who can operate the Saw. If the Operation
will take 3 days, it is unlikely that operator will be willing to work those 3 days
straight (the Saw probably won't mind). If the Site runs three shifts, and there are
operators in each shift that are capable of operating the Saw, then a ONE Load for that
'skill' will select one operator Resource during each shift. During the 3 days, as many
as 9 different Resources could be employed by that single ONE Load.

The model has selectors:
usage_policy.

The Load model has these submodels:
Load_Size.

The Load model has fields that references these models:
Resource, Skill, Load_Size, Location, Operation.

These models have a field that is a Load model:
Operation, Load_Plan, Load_Size.

Model	Load Model
-------	------------

The key field for this model is name

name - - *a Symbol field of model Load*

The name for this load. In many cases, a reasonable name for the load is resource.name or skill.name. If the operation puts more than one load on the resource or skill, the names must differ.

Default: None -- this is a key field

usage_policy - - *an extension selector of model Load*

Specifies how Resources are loaded by this Operation. The default is RESOURCE, which simply loads the specified 'resource'. Common alternatives are: ONE, which loads one of the Resources with 'skill'; MAX, which loads 'max_resources', or all available, of Resources with 'skill'; MIN, MAX, which loads at least 'min_resources', and up to 'max_resources' of 'skill'. Each of the latter two allows you to specify a single 'efficiency_loss' that occurs for each added Resource. Other extensions can provide more precise specifications of efficiency loss, including a table of values for each number of Resources.

Default: RESOURCE

Extensions:

RESOURCE, ONE,

resource - - *a Resource field of model Load*

The Resource required for the 'owner' Operation. This Resource will be loaded for the duration of the Operation.

If the 'usage_policy' is not RESOURCE, setting this field will also set 'usage_policy' to RESOURCE and will set the 'skill' field to [unspecified] (you can either specify a Skill or a Resource, but not both).

Default: [unspecified]

skill - - *a Skill field of model Load*

The Skill required for the 'owner' Operation. A Resource capable of this Skill must be chosen and prepared to perform this Skill. The chosen Resource will be loaded for the duration of the Operation.

If the 'usage_policy' is RESOURCE, setting this field will also set 'usage_policy' to ONE and will set the 'resource' field to [unspecified] (you can either specify a Skill or a Resource, but not both).

Default: [unspecified]

Model	Load Model
-------	------------

load_sizes - - *a list of Load_Size submodels of model Load*

Load_Size represents one dimension of the size of this Load. The Load may have zero, one or more dimensions of size. Each dimension will have a name and a size (in Quantity). For example, a 'box' load may have a 'weight' dimension, a 'volume' dimension, a 'Height' dimension, a 'width' dimension and so on. These sizes are used by the resources used by this load.

start_setup - - *a Symbol field of model Load*

The required starting setup of the Resource for the 'owner' Operation. If none, then no particular start_setup is required.

Default: none

end_setup - - *a Symbol field of model Load*

The resulting setup state of the Resource following the 'owner' Operation. If none, then this Operation can leave the Resource in any desired setup. This is only appropriate for setup Operations.

Default: none

start_location - - *a Location field of model Load*

The required starting Location of the Resource for the 'owner' Operation. If [unspecified], then no particular Location is required.

Default: [unspecified]

end_location - - *a Location field of model Load*

The resulting Location of the Resource following the 'owner' Operation. If [unspecified], then this Operation can leave the Resource in any Location. That is appropriate for any fixed-Location Resource, or transit Operations.

Default: [unspecified]

owner - - *a Operation field of model Load*

The Operation that places this Load on a Resource capable of the 'skill'.

Properties: Export-Only Field

Model	Load_Size Model
-------	-----------------

5.1.1.8.1.1 Load_Size Model

Load_Size -- a submodel of model Load

Load_Size represents one dimension of the size of this Load. The Load may have zero, one or more dimensions of size. Each dimension will have a name and a size (in Quantity). For example, a box_load may have a "weight" dimension, a "volume" dimension, a "Height" dimension, a "width" dimension and so on. These sizes are used by the resources used by this load.

The model has selectors:
load_size_usage.

The Load_Size model has fields that references these models :
Load.

These models have a field that is a Load_Size model :
Load.

The key field for this model is name

name -- a Symbol/field of model Load_Size
The name of this load_size
Default: None -- this is a key field

load_size_usage -- an extension selector of model Load_Size
Represents consumption of the resource in this dimension A FIXED load_size_usage always consumes the same 'fixed_quantity' of the resource irrespective of the units of the op_plan, whereas a LINEAR load_size_usage consumes 'linear_quantity' of the resource per unit of the op_plan. The default is LINEAR.

Default: LINEAR
Extensions:
FIXED, LINEAR.

owner -- a Load field of model Load_Size

Properties: Export-Only Field

Model	Flow Model
-------	------------

5.1.1.8.2 Flow Model

Flow -- a submodel of model Operation

These are the items that are produced or consumed by this Operation. Each may have a different extension -- some may have yield, some may not -- etc. The yield applied to a particular Flow_Plan is just the yield during the Flow_Plan, not during the Operation as a whole.

The model has selectors:
usage_policy.

The Flow model has fields that references these models :
Buffer, Operation.

These models have a field that is a Flow model :
Operation, Flow_Plan.

The key field for this model is name

name -- a Symbol/field of model Flow
The name for this flow. In many cases, a reasonable name for the load is buffername appended onto "PRODUCE" or "CONSUME". If a buffer has more than one PRODUCE or CONSUME flow in the operation, add something to differentiate the names.
Default: None -- this is a key field

buffer -- a Buffer field of model Flow
The Buffer id/from which this Flow of Items will occur.

If the 'buffer' is [unspecified] (or if 'buffer:phantom' is "true"), then this Flow will behave as a phantom flow (see the 'phantom' field) and not actually be posted to any Buffer.
Default: NULL

phantom -- a Logical field of model Flow
If "true", then this Flow is not actually posted to the 'buffer'. Thus, the existence of this Flow is ignored unless used by the super Operation (for example, a ROUTING Operation can use matching phantom Flows between its sub_operations to determine relative Quantities).

Models	Flow Model
--------	------------

This field is particularly useful when engineering bills or existing bills specify an intermediate item/flow that manufacturing does not want to stock, or where manufacturing has chosen to combine routings together to reduce overhead.

It is also useful for specifying yields for each of the Operations within a routing, even if the intermediate materials are of no interest. In that way, routings can properly compensate for yield loss by increasing the quantity of each upstream sub_operation.

Note that even if this is "false", if buffer.phantom is "true" or 'buffer' is [unspecified], then this Flow will behave as a Phantom Flow.

Refer to the phantom flow_policy extension of model buffer for more details.

Default: false

produced - - a Logical field of model Flow

If "true", then the 'owner' Operation is producing items into the buffer. If "false", then the 'owner' Operation is consuming items from the buffer. This is specified by the flow's usage_policy and related fields.

Properties: Export-Only Field

usage_policy - - an extension selector of model Flow

Specifies the way this item is consumed/produced by this Operation.

Default: CONSUME_PER

Extensions:

CONSUME_FIXED, CONSUME_PER, PRODUCE_FIXED, PRODUCE_PER, PRODUCE_YIELD, PRODUCE_YIELD_CALENDAR,

owner - - a Operation field of model Flow

Properties: Export-Only Field

Models	usage_policy submodels of model Flow
--------	--------------------------------------

5.1.1.8.2.1 usage_policy submodels of model Flow

5.1.1.8.3 Operation_Problem_Detector Model

Operation_Problem_Detector -- a submodel of model Operation

problem_detection Field A list of additional Problem Detectors that are applied to this Operation. They are notified on each change to quantity or dates of an Operation_Plan such that they can identify specific Problems.

Such Problems are resolved in a fairly disintegrated fashion compared to a 'process' designed to deal with the same issues, but the Problems are accurately identified. Some may require manual aid to be resolved well. Some will work well automated with certain 'process's, but not with others.

The primary intent is to provide full flexibility and orthogonality in describing all of the Problems that need to be detected, even if an intelligent solver (in the form of a 'process' extension) has not been provided for that particular combination of Problems.

For example, there may be no 'process' that Despite that, we may have individual Operation_Problem_Detectors that can deal with each one of those, individually. By just dropping each of those into this list, independently, all of those Problems will be accurately detected, supporting full 'visibility'. However, there would be no algorithm designed for intelligently planning with that specific combination of four 'problem_detectors'.

The model has selectors:
detector.

The Operation_Problem_Detector model has fields that references these models :
Operation.

These models have a field that is a Operation_Problem_Detector model :
Operation.

The key field for this model is detector

detector -- an extension selector of model Operation_Problem_Detector
The problem to be detected, and possibly resolved.
Default: None -- this is a key field
Extensions:

feasible -- a Logical field of model Operation_Problem_Detector
If 'false', then this Problem represents an infeasibility that must be resolved to make the plan feasible. In that case, this will have infinite 'cost'.

Setting this from 'true' to 'false' also sets 'cost' to 'oo'. Setting this from 'false' to 'true' also sets 'cost' from 'oo' to '0'. Note that this field is strictly unnecessary -- 'cost' is sufficient -- this is provided purely for convenience.

Default: false

cost -- a Money field of model Operation_Problem_Detector

The cost incurred if this Problem is not resolved. If infinite, then this Problem represents an infeasibility that must be resolved to make the plan feasible. In that case, the field 'feasible' will be 'false'.

Setting this to "oo" also sets 'feasible' to 'false'. Setting this to a finite value also sets 'feasible' to 'true'.
Default: oo

owner -- a Operation field of model Operation_Problem_Detector

Properties: Export-Only Field

Models	process submodels of model Operation
--------	--------------------------------------

5.1.18.4 process submodels of model Operation

5.1.18.5 Alternate_Operation Model

Alternate_Operation -- *a submodel of model Operation*

One alternate 'operation' that can be selected for performing the 'owner' Operation.

The Alternate_Operation model has fields that references these models :
Operation.

These models have a field that is a Alternate_Operation model :
ALTERNATES_PRIMARY, ALTERNATES_MANUAL,
ALTERNATES_PROPORTIONAL, ALTERNATES_PREFERENCE.

The key field for this model is operation

operation -- *a Operation field of model Alternate_Operation*

The Operation to be performed as a 'sub_operation' of the 'owner' Operation. The same Operation may not appear in the 'owner' Operation multiple times. Operation. Setting this field will cause existing operation plan trees using this alternate to be modified to be consistent with the new value, but the exact effect on those operation plans is undefined. They may be deleted or moved to another alternate.

Default: None -- this is a key field

quantity_per -- *a Quantity field of model Alternate_Operation*

The Quantity of 'operation' that must be performed per unit of the 'owner' Operation. The Quantity is converted into units of 'operation'. And, thus, unless Quantity's, such as the default value of "1", means that number of units of 'operation'. In the case of the default, one unit of 'operation' is performed for each unit of the 'owner' Operation. Setting this field will cause existing operation plan trees using this alternate to be modified to be consistent with the new value, but the exact effect on those operation plans is undefined. They may be resized, deleted, or moved to another alternate.

Default: 1

rank -- *a Integer field of model Alternate_Operation*

The rank is used by SEQUENTIAL_ALTERNATE strategy (see execution extension of strategy). The lower number has higher priority.

Properties: Export-Only Field

percentage -- *a Percentage field of model Alternate_Operation*

This may be the percentage of times this alternate 'operation' is typically selected, percentage it should be selected, or the priority of selection.

Models	Effective_Calendar_Operation Model
--------	------------------------------------

Default: 100%

owner -- *a Operation field of model Alternate_Operation*
The Operation that will perform this 'operation' as one of its 'sub_operations'.
Properties: Export-Only Field

5.1.18.6 Effective_Calendar_Operation Model

Effective_Calendar_Operation -- *a submodel of model Operation*

One effective alternate 'operation' that can be selected for performing the 'owner' Operation.

The Effective_Calendar_Operation model has fields that references these models :
Operation, Calendar.

These models have a field that is a Effective_Calendar_Operation model :
EFFECTIVE_CALENDAR.

The key field for this model is operation

operation -- *a Operation field of model Effective_Calendar_Operation*

The Operation to be performed as a 'sub_operation' of the 'owner' Operation. The same Operation may not appear in the 'owner' Operation multiple times.

Default: None -- this is a key field

quantity_per -- *a Quantity field of model Effective_Calendar_Operation*

The Quantity of 'operation' that must be performed per unit of the 'owner' Operation. The Quantity is converted into units of 'operation'. And, thus, unless Quantity's, such as the default value of "1", means that number of units of 'operation'. In the case of the default, one unit of 'operation' is performed for each unit of the 'owner' Operation.

Default: 1

percentage -- *a Percentage field of model Effective_Calendar_Operation*

This may be the percentage of times this alternate 'operation' is typically selected, percentage it should be selected, or the priority of selection.

Default: 100%

calendar -- *a Calendar field of model Effective_Calendar_Operation*

This specifies the effectivity calendar for the 'operation'. Each calendar entry should specify the name of the operation that is effective at that time.
Default: [unspecified]

Models	Routing_Operation Model
--------	-------------------------

owner -- a *Operation field of model Effective_Calendar_Operation*
 The Operation that will perform this 'operation' as one of its sub_operations.
 Properties: Export-Only Field

5.1.1.8.7 Routing_Operation Model

Routing_Operation -- a *submodel of model Operation*

A sequenced sub_operation of the 'owner' Routing_Operation.

The Routing_Operation model has fields that references these models :
 Operation.

These models have a field that is a Routing_Operation model :
 ROUTING, TRANSFER_ROUTING, FLOW_ROUTING.

The key field for this model is sequence_number

sequence_number -- a *Number field of model Routing_Operation*
 The smaller the Number, the earlier in the sequence it is placed. The
 'sequence_number's within the 'owner' Operation must be unique.

Default: None -- this is a key field
 Properties: required=true

operation -- a *Operation field of model Routing_Operation*

The Operation to be performed as a 'sub_operation' of the 'owner' Operation. The same
 Operation may appear in the 'owner' Operation multiple times (with different
 'sequence_number's of course).
 Default: [unspecified]

quantity_per -- a *Quantity field of model Routing_Operation*

The Quantity of 'operation' that must be performed per unit of the 'owner' Operation.
 The Quantity is converted into units of 'operation'. And, thus, unitless Quantity's, such
 as the default value of '1', means that number of units of 'operation'. In the case of the
 default, one unit of 'operation' is performed for each unit of the 'owner' Operation.

If this value is less than or equal to 0, then this value is ignored. In that case, the Oper-
 ation must have a Flow complementary to the successive sub_operation, and the quan-
 tity will be determined based upon that. Given the variety of Flow_usage_policy's, this
 approach offers much more flexibility. In particular, for modeling yield's and similar
 complexities. Note that such Flows may be phantom's, or the Buffers they flow to and
 from may be phantom's; in either case, the Flows will not be posted to a Buffer_Plan
 or result in any other planning structures.

Models	Configuration Model
--------	---------------------

Default: 1

owner -- a *Operation field of model Routing_Operation*
 The Operation that will perform this 'operation' as one of its sub_operations.
 Properties: Export-Only Field

5.1.1.9 Configuration Model

Configuration -- a *submodel of model Site*

A Configuration models an Item with specific desired characteristics. For example, an
 OPTIONED Item cannot be built without specifying the desired options. Such infor-
 mation is specified in this, the Configuration.

The model has selectors:
 spec.

The Configuration model has fields that references these models :
 Item, Site.

These models have a field that is a Configuration model :
 LINK, Item_Request, Item_Acceptance, Item_Promise, Lot, PRODUCE,
 CONSUME, DELIVER, SIMPLE_FIXED_QUANTITY,
 SINGLE_REQUEST, SIMPLE_FIXED_TIME, WEEKLY,
 DUAL_REQUEST, REQUEST_FIXED,
 REQUEST_FIXED_WITH_ANALYSIS, Option_Spec.

The key field for this model is item

item -- a *Item field of model Configuration*

The Item for which this Configuration specifies desired characteristics.

Default: None -- this is a key field

spec -- an *extension selector of model Configuration*

The value of this 'spec' extension selector is defined by 'item.spec'. It defines the fields
 of this Configuration needed to specify the Item precisely enough to define inter-
 changeability.

For instance, an OPTIONED Configuration specifies desired options for various fea-
 tures. The options specified in two Configurations of an OPTIONED Item must match
 to be interchangeable. A STANDARD Configuration has no specification fields -- they
 are all interchangeable. In contrast, a CUSTOM Configuration also has no specifica-
 tion fields, but they are all considered distinct (not interchangeable).

Model	Configuration Model
Properties: Extensions: STANDAARD, CUSTOM,	Export-Only Field
Interchangeable (Configuration) -- a Logical field of model Configuration	
If "true", then Items of the argument Configuration are interchangeable with Items of this Configuration. They have equivalent characteristics of those that determine interchangeability.	
Properties:	Export-Only Field
owner -- a Site field of model Configuration	
The Item for which this Configuration specifies the desired characteristics. This is the same as Item.	
Properties:	Export-Only Field

5.1.1.9.1 spec submodels of model Configuration
5.1.1.10 Item_Group Model
Item_Group -- a submodel of model Site

A Item_Group models a hierarchical grouping or classification of Items. A Item can only appear directly in only one Item_Group of a hierarchy. Similarly, a Item_Group can only appear directly in one place in the hierarchy. However, a Item can appear in any number of separate Item_Group hierarchies.

An implication of this is that all the Item_Groups cannot be assembled into the one (unspecified) Item_Group -- the hierarchies must remain separate, otherwise the totals for one Item would be counted dozens of times or the totals would not add in an understandable way.

For example, milk Items may be grouped by fat content: "skin", "1%", "2%", and "whole". The same milk Items may also be grouped by size: pint, quart, half-gallon, and gallon. Further, the same milk Items may be grouped by brand: "Econo-Cow" and "Pure-White". A particular Item would be in one fat content group, one size group, and one brand group.

Item_Groups are organized into hierarchies. In the above example, the four fat groups might be in the "Milk by Fat" Item_Group, the four sizes might be in the "Milk By Size" group, and the two brands might be in the "Milk By Brands" group. Further, if there are non-milk Items, those groups may be included into other Item_Groups. For instance, "Milk By Size" may be in the "By Drink By Size" Item_Group, which also includes "Orange_Juice_By_Size" and others.

Note that grouping "Milk by Fat", "Milk by Size", and "Milk by Brands" into a "Milk" group would result in counting each Milk item as many as three times. Each "Milk by *" group gives the summary of all Milk, but then breaks it out different ways.

All of the above milk Item_Groups are related to Item characteristics. You can also form Item_Groups based on other properties of the Items. For instance, your Items may be divided based upon groups of customers (Site_Groups), or "market channels", or "industry segments". For example, pints of milk sold to grocery stores may be a separate Item from pints of milk sold to restaurants. A Item_Group may be formed based on that division: "Milk By Channel".

Another common division will be by delivery lead time. Often the same Item will be made available as one Item with a 2-week delivery lead time and another with a 6-week delivery lead time, where the latter has a significantly lower price. When Items are so divided, they will almost always be grouped together for forecast purposes.

The Item_Group model has these submodels :

Sub_Item_Group, Sub_Item.

The Item_Group model has fields that references these models :
Item_Group, Sub_Item_Group, Site.

These models have a field that is a Item_Group model :
LINK, Item_Group, Sub_Item_Group, Sub_Item.

The key field for this model is name
This model may be extended with user-defined fields.

name -- *a Symbol field of model Item_Group*

The name of this Item_Group. This name must be unique among Item_Groups of the owner Site.

Default: None -- this is a key field

description -- *a String field of model Item_Group*

A description of this Item_Group.

Default: none

top -- *a Logical field of model Item_Group*

If "true", then this Item_Group appears in no other Item_Groups -- it is the top of a Item_Group hierarchy.

Properties: Export-Only Field

root -- *a Item_Group field of model Item_Group*

The root of this Item_Group tree.

Properties: Export-Only Field

sub_groups -- *a list of Sub_Item_Group submodels of model Item_Group*

The Item_Groups that are direct members of this Item_Group. Note that no Item_Group descendant can contain this Item_Group or contain common Items.

sub_items -- *a list of Sub_Item submodels of model Item_Group*
The Items that are direct members of this Item_Group. This does not include the Items in this Item_Group's sub_items. The all_items field provides a complete list.

all_items, all_items (List(Item)) -- *a List(Item) field of model Item_Group*
All of the Items that are in this Item_Group, both directly and indirectly. This list includes the Items in sub_items plus all_items of each of the Item_Groups in sub_groups.

If passed List(Item), it returns only those Items that are in the passed List and in this Item_Group. In other words, it returns the intersection of the Items in this Item_Group and the argument List.

Properties: Export-Only Field

owner -- *a Site field of model Item_Group*

The Site to which this Item_Group belongs.

Properties: Export-Only Field

5.1.1.10.1 Sub_Item_Group Model

Sub_Item_Group -- a submodel of model Item_Group

The Item_Groups that are direct members of this Item_Group. Note that no Item_Group descendant can contain this Item_Group or contain common Items.

The Sub_Item_Group model has fields that references these models :
Item_Group.

These models have a field that is a Sub_Item_Group model :
Item_Group.

The key field for this model is item_group

item_group -- a Item_Group field of model Sub_Item_Group

A Item_Group that is a direct member of the owner Item_Group.

Default: None -- this is a key field

owner -- a Item_Group field of model Sub_Item_Group

Properties: Export-Only Field

5.1.1.10.2 Sub_Item Model

Sub_Item -- a submodel of model Item_Group

The Items that are direct members of this Item_Group. This does not include the Items in this Item_Group's 'sub_items'. The 'all_items' field provides a complete list.

The Sub_Item model has fields that references these models :
Item, Item_Group.

These models have a field that is a Sub_Item model :
Item_Group.

The key field for this model is item

item -- a Item field of model Sub_Item

A Item that is a direct member of the owner Item_Group.

Default: None -- this is a key field

owner -- a Item_Group field of model Sub_Item

Properties: Export-Only Field

Models	Seller Model
--------	--------------

5.1.2 Seller Model

Seller -- a submodel of model Supply_Chain

A Seller can model a sales person, group, channel, territory, or organization. It represents the responsibility for forecasting demand, committing to sales, managing allocations, taking orders, and promising orders.

Sellers may take Requests from one Site or multiple Sites for items supplied by one Site or several different Sites. It manages the Requests and Promises made between those Sites. A Seller can act as an agent of the supplying Site and make Promises for those Sites.

Sellers can manage Requests and Promises for one Product or many Products. Products can be defined for certain customer(s), certain item(s), certain Order Lead Time, certain Price, and so on. Each Product can be Forecasted independently, or grouped with other Products into Product_Groups.

Sellers can form a hierarchy. Each Seller can be a member of another Seller, its 'organization'. Allocations can be made to any level in the hierarchy. Sellers can use allocations to themselves or any of their organizations.

The model has selectors:
request_naming.

The Seller model has these submodels :
Site_Group, Product_Root, Product, Product_Group.

The Seller model has fields that references these models :
Seller, Site_Group, Product_Root, Product_Group, Horizon, Seller_Plan, Supply_Chain.

These models have a field that is a Seller model :
Supply_Chain, Seller, Product_Root, Product, Seller_Plan, Site_Group, Product_Group, Product_Allocation, REQUEST_FIXED, REQUEST_FIXED_WITH_ANALYSIS.

The key field for this model is name
This model may be extended with user-defined fields.

name -- a Symbol field of model Seller
Full name of this Seller. It must be unique among Sellers in the owner's Supply_Chain.

Models	Seller Model
--------	--------------

Default: None -- this is a key field

description -- a String field of model Seller
A description of this Seller.

Default: none

category -- a Symbol field of model Seller
A simple user-defined categorization of Sellers. This is often used in reports. This will allow users to filter the Sellers by category, and to write expressions that depend upon the category of the Seller.

The standard reports look for the values "Office", "Channel", and "Customer". "Customer" indicates that the Seller models sales broken out by customer. "Channel" indicates that the Seller models sales broken out by market segment. "Office" indicates that the Seller models sales broken out by division, sales office, or store front.

Default: Channel

sub_category -- a Symbol field of model Seller

A simple user-defined categorization of Sellers. This is often used in reports. This will allow users to filter the Sellers by category, and to write expressions that depend upon the category of the Sellers.

The standard reports don not look for any specific values. But any values can be used - whatever is useful for categorization in reports.
Default: [unspecified]

rank -- a Number field of model Seller
Rank of this seller. This rank is used to prioritize the allocations. Sellers with higher rank get their allocations before it goes to lower rank sellers. If allocated amount is less than what is needed by sellers it is distributed proportionally among the sellers as specified by MEMBER_RANK allocation policy. Higher the number, higher the rank is.
Default: 0

organization -- a Seller field of model Seller
The Seller organization that this Seller is a member of. A Seller 'organization' can represent a Sales Office, Sales Organization, Sales Manager, Market Channel, Sales Territory, etc.

The Products and Product_Groups of the 'organization' are used as if they are Products and Product_Groups of this Seller. This Seller can have Products and Product_Groups that are not used by its 'organization'.

Models	Seller Model
--------	--------------

Default: [unspecified]

members - - - a List(Seller) *field of model Seller*

This List contains each Seller that specifies this Seller as its 'organization'. If this List is empty, then this Seller is not an 'organization' for any other.

Properties: Export-Only Field

member_of (Seller) - - - a Logical *field of model Seller*

Returns "true" if this Seller or any of its 'organization's is the parameter Seller. More precisely, if this Seller is the parameter Seller, then it returns "true"; otherwise, if this Seller's 'organization' is [unspecified], then it returns "false"; otherwise, it returns 'organization.member_of(parameter)'.
Properties: Export-Only Field

site_groups - - - a list of Site_Group *submodels of model Seller*

Each Site_Group defines a List of Sites. That list can be specified explicitly, or defined by specifying common characteristics (all Sites matching those characteristics are included in the Group). Such matching criteria can use user-defined fields of Site as well, allowing arbitrary grouping and categorization fields to be used with Sites.

Any Site_Groups defined here will appear in all 'site_groups' of this Seller and of any of its members.

all_site_groups - - - a List(Site_Group) *field of model Seller*

This List of Site_Groups includes all 'site_groups' plus the Site_Groups defined by all of this Seller's 'organization's (e.g., 'organization.organization', etc.). Thus, several Sellers can share Site_Groups by defining them in a common Seller 'organization'.

If a Seller defines a Site_Group with the same name as an 'organization's Site_Group, then the 'organization's Site_Group will not appear in this List. Only one Site_Group with a given name will appear, the one defined lowest in the Seller hierarchy. Thus, a Seller can effectively override any Site_Group of its organization.

This List is not affected by any 'members' of this Seller.

Properties: Export-Only Field

site_group (Symbol) - - - a Site_Group *field of model Seller*

The Site_Group defined by this Seller or one of its 'organization's with the specified name.

Models	Seller Model
--------	--------------

Note that 'site_group(name)' is equivalent to 'all_site_groups.find(name)', but is simpler, clearer, and more efficient.

Properties: Export-Only Field

product_roots - - - a list of Product_Root *submodels of model Seller*

The Product_Roots defined by this Seller. Each of these defines a Product-Seller tree rooted at this Seller. For each Product_Root, a Product will be created in the 'products' of this Seller and all of its members. Those Products form the Product-Seller tree.

products - - - a list of Product *submodels of model Seller*

The Products sold by this Seller. A Product is created for each Product_Root in this Seller and each of its 'organization's.

top_products - - - a List(Product) *field of model Seller*

This List is a subset of all_products consisting only of Products that are the top of a Product hierarchy, the Products that do not appear within any Product_Group.

Properties: Export-Only Field

active_products - - - a List(Product) *field of model Seller*

This List is a subset of all_products consisting only of Products that have been 'activated' (see Product.active).

Properties: Export-Only Field

product_groups - - - a list of Product_Group *submodels of model Seller*

Product_Groups group Products into hierarchies. Each Product can appear in at most one Product_Group in a hierarchy of Product_Groups. But each Product can appear in any number of independent Product_Group hierarchies.

The purpose of such hierarchical grouping is to allow forecasts to be meaningfully aggregated from Product forecasts and then adjusted and redistributed down the hierarchy to the Products. If the Product_Groups in a hierarchy contain common Products, then such aggregation is difficult to interpret or redistribute.

Any Product_Groups defined here will appear in 'product_groups' of this Seller's 'members'.

top_product_groups - - - a List(Product_Group) *field of model Seller*

This List is a subset of product_groups consisting only of Product_Groups that are the top of a Product_Group hierarchy, the Product_Groups that do not appear within any other Product_Group.

Properties: Export-Only Field

Models

Seller Model

forecast_horizon - - *a horizon field of model Seller*

Defines how the Seller_Plan's forecast_horizon is computed. This defines the lowest granularity of the forecasts for each of the Products. Each member Seller can define a finer granularity (smaller Date_Ranges), but not coarser. The forecast_horizon that results in the Seller_Plan will include all breaks specified by this horizon definition, as well as all breaks specified by any organization's fields. In a sense, the horizons are laid over the top of each other.

Default: [unspecified]

first_day_of_week - - *a Integer field of model Seller*

Monday is day 1; Sunday is day 7.

This field is obsolete and no longer be available after the 3.05 release. Use forecast_horizon.first_day_of_week instead.

Default: 1

Properties: obsolete=True

week_periods - - *a Number field of model Seller*

The number of week periods. After that number of week periods, monthly periods begin. If nonexistent, then the entire horizon is broken into weeks.

This field is obsolete and no longer be available after the 3.05 release. Use forecast_horizon.week_periods instead.

Default: nonexistent

Properties: obsolete=True

request_naming - - *an extension selector of model Seller*

Defines how names are generated for the forecast Requests generated by this Seller.

Default: NUMERIC

Extensions:

NONE.

alp_horizon - - *a horizon field of model Seller*

Defines how the Seller_Plan's alp_horizon is computed. This defines the lowest granularity of the forecasts for each of the Products. Each member Seller can define a finer granularity (smaller Date_Ranges), but not coarser. The alp_horizon that results in the Seller_Plan will include all breaks specified by this horizon definition, as well as all breaks specified by any organization's fields. In a sense, the horizons are laid over the top of each other.

Default: MONTHS

Models

Seller Model

seller_plan (Plan) - - *a Seller_Plan field of model Seller*

The Seller_Plan for this Seller in the specified Plan.

Properties: Export-Only Field

owner - - *a Supply_Chain field of model Seller*

The Supply_Chain for which this is a Seller.

Properties: Export-Only Field

Models	Site_Group Model
--------	------------------

5.1.2.1 Site_Group Model

Site_Group -- a submodel of model Seller

A Site_Group models a list of Sites, specified in one of several different ways. It may be specified explicitly by simply naming the Sites that belong in the Site_Group. Alternatively, it may specify a filter expression which computes whether a Site is included or not. Or it may look for a certain category or set of category values in the Sites.

This is commonly used to specify which Sites are customers of a particular Product.

The model has selectors:
spec.

The Site_Group model has these submodels :
Explicit_Site.

The Site_Group model has fields that references these models :
Explicit_Site, Seller.

These models have a field that is a Site_Group model :
Seller, Product_Root, Product, Explicit_Site.

The key field for this model is name
This model may be extended with user-defined fields.

name -- a Symbol field of model Site_Group
The name of this Site_Group. This name' must be unique among the Site_Groups of the 'owner' Seller.
Default: None -- this is a key field

description -- a String field of model Site_Group
A description of this Site_Group.
Default: none

sites, sites (List(Site)) -- a List(Site) field of model Site_Group
The Sites that are in this Site_Group. If passed List(Site), it returns only those Sites that are in the passed List and in this Site_Group. That is, it returns the intersection of the Sites in this Site_Group and the argument List.
Properties: Export-Only Field

Models	Site_Group Model
--------	------------------

spec -- an extension selector of model Site_Group

How the list of Sites that make up the Site_Group are specified. The default is EXPLICIT, where each Site in the group is named explicitly. Future extensions will include filtering extensions that let you filter the sites by 'category', 'name', or any number of other fields, or even an arbitrary Expression which computes whether a Site belongs. Currently, only EXPLICIT is supported.
Default: EXPLICIT

explicit_sites -- a list of Explicit_Site submodels of model Site_Group
Identifies a Site, 'site', that is explicitly included in the 'owner' Site_Group.

owner -- a Seller field of model Site_Group
The Seller to which this Site_Group belongs.
Properties: Export-Only Field

5.1.2.1.1 Explicit_Site Model

Explicit_Site -- a submodel of model Site_Group

Identifies a Site, 'site', that is explicitly included in the 'owner' Site_Group.

The Explicit_Site model has fields that references these models :
Site, Site_Group.

These models have a field that is a Explicit_Site model :
Site_Group.

The key field for this model is site

site -- a Site field of model Explicit_Site

The Site that has been explicitly added to this Site_Group.

Default: None -- this is a key field

owner -- a Site_Group field of model Explicit_Site

Properties: Export-Only Field

5.1.2.2 Product_Root Model

Product_Root -- a submodel of model Seller

Product_Root defines the base information of a Product. Defining a Product_Root defines a Product_Seller tree. The root of that tree is the Product in this Seller. The corresponding Products in each of the 'members' of this Seller form the rest of the Product_Seller tree. Products cannot be added directly -- a Product_Root must be added.

A Product defines one or more Items that are available to a set of customers with a certain delivery lead time to certain delivery territories, at a certain price. Any of those may be unlimited (any delivery lead time; any delivery address etc.). Each Product is independently forecasted, allocated, and priced for the purposes of quoting/promising. See the Product model for more information on Products.

The Product_Root defines...

In each Seller, there is a special uneditable Product_Root named [unspecified]. It has the default values for all fields. Since its Item is [unspecified], this Product is not real and cannot be actually requested, promised, or planned.

The model has selectors:
forecast_policy.

The Product_Root model has these submodels :

Product_Supplier.

The Product_Root model has fields that references these models :

Product_Supplier, Site, Site_Group, Unit, Seller.

These models have a field that is a Product_Root model :

Seller, Product, Product_Supplier, BUCKETED_NESTED_SORT.

The key field for this model is name

This model may be extended with user-defined fields.

name - - *a Symbol/field of model Product_Root*

The name of this Product(_Root). Depending upon the environment, this is typically the name of the Item (Item family, or pseudo item), name of the Buffer (Buffer family, or SKU), or the name of the Operation. However, it is also often none of those. It simply must be unique among the Products of the 'owner' Seller, but otherwise can be whatever is meaningful.

Default: None -- this is a key field

description - - *a String/field of model Product_Root*
A description of this Product(_Root).

Default: none

suppliers - - *a list of Product_Supplier submodels of model Product_Root*

The Items that are sold as this Product(_Root), organized by supplying Site. Only Item_Requests for these Items may be filled by this Product(_Root). Forecasts for this Product(_Root) only include sales of these Items.

min_quantity - - *a Quantity/field of model Product_Root*

An Item_Request with a quantity less than this cannot be filled by this Product(_Root). Forecasts for this Product(_Root) only include sales with this and larger Quantity. This Quantity is converted to the unit of this Product(_Root).

Default: 0

min_delivery_lead_time - - *a Time/field of model Product_Root*

An Item_Request with an delivery lead time less than this cannot be filled by this Product(_Root). Forecasts for this Product(_Root) only include sales with this and larger delivery lead time. Delivery lead time is the Time from when the Promise must be accepted ('accept_by') to when the delivery must be made ('due_end').

Default: 0

delivery_area - - *a Symbol/field of model Product_Root*

Defines the delivery area that this Product(_Root) covers. Forecasts for this Product(_Root) only include sales within this delivery area.

Default: [unspecified]

effective_dates - - *a Date_Range/field of model Product_Root*

A Request for this Product can be filled during these dates.

Default: infinite date range

customer - - *a Site/field of model Product_Root*

Item_Requests from this customer Site may be filled by this Product(_Root).

The fields 'customer' and 'customers' both specify the customer Sites that may be filled by this Product(_Root). The 'customers' field is a Site_Group that can contain a List of Sites. It is the general mechanism for specifying allowed customer Sites. For convenience, the field 'customer' can be used for specifying a single Site. It can be used without needing to define a Site_Group, and thus is simpler to use. These two fields are additive: effectively the 'customer' is appended to the List in 'customers'. Both can be specified; or either can be left [unspecified]; or both can be left [unspecified], which allows any customer Site.

Forecast for this Product(_Root) should include only sales to Sites in 'customers' or 'customer'.

Default: [unspecified]

customers - - *a Site_Group/field of model Product_Root*

Item_Requests from the Sites in this Site_Group may be filled by this Product(_Root).

The fields 'customer' and 'customers' both specify the customer Sites that may be filled by this Product(_Root). The 'customers' field is a Site_Group that can contain a List of Sites. It is the general mechanism for specifying allowed customer Sites. For convenience, the field 'customer' can be used for specifying a single Site. It can be used without needing to define a Site_Group, and thus is simpler to use. These two fields are additive: effectively the 'customer' is appended to the List in 'customers'. Both can be specified; or either can be left [unspecified]; or both can be left [unspecified], which allows any customer Site.

Forecast for this Product(_Root) should include only sales to Sites in 'customers' or 'customer'.

Default: [unspecified]

forecast_policy - - *an extension selector of model Product_Root*

Defines what forecast information will be kept in the Seller_Plan's Forecast_Entry's, how forecast Requests for this Product_Root should be computed from the committed Forecast such that they are representative of the expected demand, and how those forecast Requests should be adjusted as actual Requests arrive (or fail to arrive).

For example, it can define how to handle forecast error and timing variance, service level and safety level adjustments, end-of-month skewed forecast Requests, or how to expire or adjust the forecast as time passes and actual orders are received (or not). It can place orders for a Configuration that is representative of all the Items included, or it can use a representative mix of the Items. It may even define that forecasted mix information should be kept in each Forecast_Entry and used to compute the representative mix.

Models	Product_Root Model
--------	--------------------

Note that if the `forecast_policy` creates `forecast_requests` based on a `representative_item`, which may be specified in a `'representative_configuration'`, then that `representative_item` **MUST** be specified in order for `forecast_requests` to be generated properly. Failure to specify the `'representative_configuration'`'s item will result in `forecast_requests` for the [unspecified] item of the [unspecified] site. (The only exception to this rule is the case in which the `Product_Root` has only one `Product_Item`. In this case, it is assumed that the single `Product_Item` is intended to be the `'representative_configuration:item'100`.)

A `forecast_policy` works within the `forecast_horizon` of the `'top_seller'`. That is, it is independent of the granularity of member `seller_forecast_horizons`.

Default: `SINGLE_REQUEST`

Extensions:

`SIMPLE_FIXED_QUANTITY`, `SINGLE_REQUEST`, `SIMPLE_FIXED_TIME`, `WEEKLY_DUAL_REQUEST`.

`product, product (Product_Allocation) -- a Product field of model`

`Product_Root`

Returns the `Product` corresponding to this `Product_Root` in the specified `Seller`. If no `Seller` is specified, then this `Seller` is used, and this `Seller's Product` corresponding to this `Product_Root` is returned.

Properties: `Export-Only Field`

`unit -- a Unit field of model Product_Root`

The `unit` defines the `Quantity` of one unit of this `Product(Root)`, whether this `Product(Root)` is `discrete` (can only exist in whole units) and what the `'preferred_measure'` for this `Product(Root)` is. It can also convert between different units of `Measure` and units of this `Product(Root)`.

For example, one unit of `Product X` may be "3 kg", "400 ml", and "\$12". It may be `'discrete'`, meaning that a need for "3 kg" will result in "6 kg" (2 units) being built. The `'preferred_measure'` may be "kg" which will cause most fields that display a `Quantity` of this `Product` to display in "kg" by default.

Note that this field is not settable -- you cannot assign it a new `Unit`. Rather, you can get the `Unit` of this `Product_Root` and set that `Unit's` fields.

Properties: `Export-Only Field`

`owner -- a Seller field of model Product_Root`

The `Seller` to which this `Product_Root` belongs.

Properties: `Export-Only Field`

Models	Product_Supplier Model
--------	------------------------

5.1.2.2.1 Product_Supplier Model

`Product_Supplier -- a submodel of model Product_Root`

The items that are sold as this `Product(Root)`, organized by supplying `Site`. Only `Item_Requests` for these items may be filled by this `Product(Root)`. Forecasts for this `Product(Root)` only include sales of these items.

The `Product_Supplier` model has these submodels :
`Product_Item`.

The `Product_Supplier` model has fields that references these models :
`Site`, `Product_Item`, `Product_Root`.

These models have a field that is a `Product_Supplier` model :
`Product_Root`, `Product_Item`.

The key field for this model is `supplier`

`supplier -- a Site field of model Product_Supplier`

A `Site` that supplies at least one of the items covered by this `Product(Root)`.
Default: `None -- this is a key field`

Items -- a list of `Product_Item submodels of model Product_Supplier`
The items of the `'supplier'` `Site` that are sold as the `'owner' Product(Root)`.

`owner -- a Product_Root field of model Product_Supplier`

Properties: `Export-Only Field`

Models	Product_Item Model
--------	--------------------

5.1.2.2.1.1 Product_Item Model

Product_Item -- a submodel of model Product_Supplier

The Items of the 'supplier' Slice that are sold as the 'owner' Product(_Root).

The Product_Item model has fields that references these models :
Item, Product_Supplier.

These models have a field that is a Product_Item model :
Product_Supplier.

The key field for this model is item

Item - - a Item field of model Product_Item

A Request for this Item can be filled as this Product(_Root).

Default: None -- this is a key field

owner - - a Product_Supplier field of model Product_Item

Properties: Export-Only Field

Models	Product Model
--------	---------------

5.1.2.3 Product Model

Product -- a submodel of model Seller

A Product defines one or more Items that are available to a set of customers with a certain delivery lead time to certain delivery territories, at a certain price. Any of those may be unlimited (any delivery lead time; any delivery address; etc.). Each Product is independently forecasted, allocated, and priced for the purposes of quoting/promising.

The primary purpose of forecasting and forecast management is to estimate the sales potentials for each of these Products, independently. The primary purpose of master planning is to determine how capacity and materials will be allocated in order to best meet the forecasts for these Products. Master planning determines how much of each Product will be available and when. Order promising is then performed in terms of those allocations to these Products. In this sense, the Products define the granularity of the Seller's master plan.

Products also define divisions for pricing, which is inherently tied to order promising (if the customer is willing to pay more, they will often find more available; similarly, if the customer is willing to wait longer, they can often get a lower price). Note, however, that the pricing rules are defined by a separate model: Seller_Product_Policies. This is because different Sellers may need to set pricing independently; but they should not be redefining the Product model. So, the pricing fields have been separated out. Thus, the Products essentially define the granularity of the Seller's price list, though the Seller's price list itself is defined by the Seller_Product_Policies models.

Note that a Customer may request a specific Item on a specific Date. That Request may be satisfied by any one of a number of Products. Each Product will have independent availability and pricing. The list of all the options for meeting the customer's Request can be presented, and the customer can choose the most satisfactory among them, based upon price and timing.

From a supply point of view, the availability of a Product is estimated based upon building a representative Configuration of an Item and delivering it to a representative Customer address. The less accurately these represent all the Items and delivery addresses included, the less accurate the availability information will be. Therefore, Promises can be padded with extra lead time to compensate for possible misrepresentation. In many cases, Products can be defined at a low enough level that availability is accurately planned and no safety padding is necessary.

Model	Product Model
-------	---------------

Products cannot be introduced directly. Instead, a Product_Root must be added. There is one Product in a Seller for each Product_Root defined by that Seller or any of its organization's. Certain Product characteristics can only be defined once for the whole Product-Seller tree -- those characteristics are defined by the fields of Product_Root and are only available for viewing through the Product.

In each Seller, there is a special uneditable Product named [unspecified]. It has the default values for all fields. Since its Item is [unspecified], this Product is not real and cannot be actually requested, promised, or planned.

The model has selectors:

forecast_policy, expiration_policy, allocation_policy, availability_policy, price_policy.

The Product model has these submodels :
Generic_Product, Alternate_Product.

The Product model has fields that references these models :
Product_Root, Site, Site_Group, Generic_Product, Alternate_Product, Seller.

These models have a field that is a Product model :
Seller, Product_Root, Generic_Product, Alternate_Product, Sub_Product, INDIVIDUAL, Product_Available_To_Promise, Product_Allocation.

The key field for this model is name
This model may be extended with user-defined fields.

product_root - - *a Product_Root field of model Product*
The Product_Root that defines the existence of this Product. Many of this Product's fields are defined by this Product_Root. For example, 'name', 'description', 'items', 'min_quantity', and 'min_delivery_lead_time' cannot be set in this Product -- they must be set through this Product_Root.
Properties: Export-Only Field

name - - *a Symbol field of model Product*
The name of this Product, which is defined by the Product_Root. This is essentially a shortcut to 'product_root.name'.

Model	Product Model
-------	---------------

Depending upon the environment, this is typically the name of the Item (Item family, or pseudo item), name of the Buffer (Buffer family, or SKU), or the name of the Operation. However, it is also often none of those. It simply must be unique among the Products of the 'owner' Seller, but otherwise can be whatever is meaningful.

Default: None -- this is a key field
Properties: Export-Only Field

rank - - *a Number field of model Product*
Rank of this product. This is used to compute the planning priority of an item_rap in active_strategy.
Default: 0

description - - *a String field of model Product*
A description of this Product, as defined by the 'product_root'. This is a shortcut to 'product_root.description'.
Properties: Export-Only Field

items - - *a List(Item) field of model Product*
The End Items that are sold as this Product. These end Items could potentially come from different sites. This List cannot be edited directly; rather, you must edit the Items' List under the appropriate Site in the 'suppliers' List of the 'product_root'.
Properties: Export-Only Field

min_quantity - - *a Quantity field of model Product*
An Item_Request with a quantity less than this cannot be filled by this Product. Forecasts for this Product only include sales with this and larger Quantity. This Quantity is converted to the unit of this Product's 'product_root'.

This field cannot be edited directly; the 'min_quantity' field of the 'product_root' defines this value.
Properties: Export-Only Field

min_delivery_lead_time - - *a Time field of model Product*
An Item_Request with an delivery lead time less than this cannot be filled by this Product. Forecasts for this Product only include sales with this and larger delivery lead time. Delivery lead time is the Time from when the Promise must be accepted ('accept_by') to when the delivery must be made ('due_end').

This field cannot be edited directly; the 'min_delivery_lead_time' field of the 'product_root' defines this value.
Properties: Export-Only Field

Models	Product Model
--------	---------------

delivery_area - - *a Symbol field of model Product*
 Defines the delivery area that this Product covers. Forecasts for this Product only include sales within this delivery area.

This field cannot be edited directly; the 'delivery_area' field of the 'product_root' defines this value.

Properties: Export-Only Field

customer - - *a Site field of model Product*
 Item_Requests from this customer Site may be filled by this Product.

The fields 'customer' and 'customers' both specify the customer Sites that may be filled by this Product. The 'customers' field is a Site_Group that can contain a List of Sites. It is the general mechanism for specifying allowed customer Sites. For convenience, the field 'customer' can be used for specifying a single Site. It can be used without needing to define a Site_Group, and thus is simpler to use. These two fields are additive: effectively the 'customer' is appended to the List in 'customers'. Both can be specified; or either can be left (unspecified); or both can be left (unspecified), which allows any customer Site.

Forecasts for this Product should include only sales to Sites in 'customers' or 'customer'.

This field cannot be edited directly; the 'min_delivery_lead_time' field of the 'product_root' defines this value.

Properties: Export-Only Field

customers - - *a Site_Group field of model Product*
 Item_Requests from the Sites in this Site_Group may be filled by this Product.

The fields 'customer' and 'customers' both specify the customer Sites that may be filled by this Product. The 'customers' field is a Site_Group that can contain a List of Sites. It is the general mechanism for specifying allowed customer Sites. For convenience, the field 'customer' can be used for specifying a single Site. It can be used without needing to define a Site_Group, and thus is simpler to use. These two fields are additive: effectively the 'customer' is appended to the List in 'customers'. Both can be specified; or either can be left (unspecified); or both can be left (unspecified), which allows any customer Site.

Forecasts for this Product should include only sales to Sites in 'customers' or 'customer'.

Models	Product Model
--------	---------------

This field cannot be edited directly; the 'min_delivery_lead_time' field of the 'product_root' defines this value.

Properties: Export-Only Field

forecast_policy - - *an extension selector of model Product*
 Defines what forecast information will be kept in the Seller_Plan's Forecast_Entry's, how forecast Requests for this Product should be computed from the 'committed' forecast such that they are representative of the expected demand, and how those forecast Requests should be adjusted as actual Requests arrive (or fail to arrive).

For example, it can define how to handle forecast error and timing variance, service level and safety level adjustments, end-of-month skewed forecast Requests, or how to expire or adjust the forecast as time passes and actual orders are received (or not). It can place orders for a Configuration that is representative of all the 'Items' included, or it can use a representative mix of the 'Items'. It may even define that forecasted mix information should be kept in each Forecast_Entry and used to compute the representative mix.

This field cannot be edited directly; the 'min_delivery_lead_time' field of the 'product_root' defines this value.

Properties: Export-Only Field

Extensions:

SIMPLE_FIXED_QUANTITY_SINGLE_REQUEST, **SIMPLE_FIXED_TIME_WEEKLY_DUAL_REQUEST**.

active - - *a Logical field of model Product*

A Product is 'active' if any of it's fields has been set to other than the default value. The conversion to "active" can occur by setting a field for a Product, by activating any of a Product's specific Products (see Product.specific_products) or by creating an active Forecast for the Product.

The default for each of the Product model fields which return List(Problem) is to return the complete set of Products (both 'inactive' and 'active'). Each such field will also have an 'active' counterpart. For example, Seller.products() returns all Products of the Sellers, but Seller.active_products() returns only active products of the Seller.

The 'active' vs 'inactive' distinction is useful in situations where the number of possible Seller/Product_Root combinations is much larger than the number of Seller/Product_Root combinations actively being forecasted.

The 'active' field is not settable.

Models	Product Model
--------	---------------

Properties: Export-Only Field

expiration_policy - - *an extension selector of model Product*

Defines how the forecast for this Seller expires as time passes if the actual demand is less than expected.

The default **AT_END** expiration_policy, for example, does not expire any of the forecast until the end of the forecast period. So, if 1000 units are forecasted, but only 10 units have actually been requested through the 20th day of the month period, the other 990 units will continue to be forecasted to be sold in the last few days.

Default: **AT_END**

Extensions:

AT_END.

allocation_policy - - *an extension selector of model Product*

Defines how allocations of this Product to the owner, Seller are allocated to the Seller's members or customers. In the absence of actual orders, this policy simply allocates to the members. With actual orders present, the supplying Sites will have made allocations directly to those actual orders. Thus, the Allocation_Policy can only report Problems when the allocations are inconsistent with its rules. Those Problems will need to be addressed along with all of the other Problems at each Site. The default allocation_policy is **PER_COMMITTED**. Setting the allocation policy on a Product that a Seller sells, does not recursively set the allocation_policy on the products that the sub sellers sell.

For example, the **PER_COMMITTED** policy distributes the allocations proportional to the quantity committed to by that Seller. The **FIXED_SPLIT** policy distributes according a fixed percentage breakdown among the members. When implemented, **CALENDAR_SPLIT** will be similar, but the splits may vary over time. The **FCFS** policy retains the whole allocation at this Seller, forcing the members to consume from this Seller first-come first-served (FCFS).

Default: **PER_COMMITTED**

Extensions:

FCFS, PER_ALLOCATED, PER_COMMITTED, MEMBER_RANK, FIXED_SPLIT.

auto_allocate - - *a logical field of model Product*

Specifies whether or not allocation to members' Sellers occurs automatically. When set to **TRUE**, changes to Forecast_Entry allocated quantities are immediately allocated to members' Sellers using this Seller's allocation_policy. When set to **FALSE**, changes to Forecast_Entry allocated quantities are not automatically allocated to members' Sellers.

Models	Product Model
--------	---------------

Default: **TRUE**

always_override_members_forecasted - - *a logical field of model Product*

Specifies whether or not forecasted quantities of member sellers are aggregated up to the seller organization. If this is "true", then the member seller quantities are not aggregated.

Default: **false**

always_override_members_committed - - *a logical field of model Product*

Specifies whether or not committed quantities of member sellers are aggregated up to the seller organization. If this is "true", then the member seller quantities are not aggregated.

Default: **false**

availability_policy - - *an extension selector of model Product*

A new extension, called 'availability', will be added. The following 2 extensions will be supported (initially), **SLIDING** and **HORIZON**. Specifies the method for computing product availability, (i.e. cumulative ATP) for this Product. The default availability policy, **SLIDING**, uses only the product.consume_carrier field in computing cumulative ATP. Other availability_policies extend the 'consume_carrier' capability by adding concepts such as fixed bounds which limit product availability beyond fixed bounds. These bounding policies may be used to model such realities as fiscal quarters.

Default: **SLIDING**

Extensions:

SLIDING, HORIZON.

price_policy - - *an extension selector of model Product*

Specifies how the price of an Item_Promise is computed. The Item, customer, delivery address, quantity, and delivery lead time are all typical factors used to compute the price. Some price_policy's use Expressions which can compute from arbitrary user-defined fields on the Products, Requests, or Promises.

Default: **FIXED**

Extensions:

NONE, FIXED.

Models	Product Model
--------	---------------

`consume_carrier` - - - *a Time field of model Product*
 The amount of time into the past (i.e. earlier) that is considered when calculating the ATP for a given actual promise. This time is measured from the ending date of each `Forecast_Entry` (i.e. `Forecast_entry.delivery_dates.end`). The traditional definition of ATP would set this at 00, meaning that an actual promise can consume from any earlier `Forecast_Entry`. However, it is not always reasonable to assume that if you can build it early, you can deliver it later.

Having the capacity to build it early does not imply the ability to build it later, either. If earlier delivery is not acceptable, that means building it early and storing it. In this case, consuming earlier ATP can result in increased storage costs. If storing the items for excessively long periods is not acceptable (e.g., perishable items), consuming earlier ATP can result in exceeding shelf life limits. Or if the delivery_operation is critical, then it may not be possible to build it early, store it, and then deliver it later.

Note that Requests can give a range of delivery Dates. Anything available during that range can be consumed, plus anything available up to `consume_carrier` than the earliest Date in the range.

For any bucket that is completely enclosed by the `Date_Range`, from the ending date of the bucket minus `consume_carrier` to the latest requested delivery Date, the whole bucket can be consumed. For example, if we have weekly buckets beginning at 5/1 and 5/8, and a delivery is requested between 5/12 and 5/14, and `consume_carrier` is set to "7 days", then the full available to promise from both the 5/1 bucket and the 5/8 bucket may be consumed. However, if `consume_carrier` is set to "4 days", then none of the 5/1 bucket may be consumed. Consider that months can be 28, 29, 30, and 31 days long. So, in some cases `consume_carrier` of 61 days will result in 2 forecast_entry buckets (for months of 28, 29, and 30 days) and only 1 bucket in other cases (for months of 31 days).

What this means is that certain values of `consume_carrier` should not be used when using a Monthly bucket_spec. That is, values between 56-61, 84-92, 112-124, etc. are problematic since the lower bound is evenly divisible by 28 and the upper bound is 1 shy of being evenly divisible by 31. So, when using a Monthly bucket_spec, `consume_carrier` values should therefore be 32, 63, 94, 125, etc. for 1, 2, 3, 4, etc. months of "consume earlier" ATP quotes.
 Default: 00

Models	Product Model
--------	---------------

`time_pad` - - - *a Time field of model Product*
 Additional Time by which the due Dates quoted to customers are padded to compensate for the mis-estimation due to a poorly representative
 representative_configuration. Each Seller can override this with its own Product model.
 Default: 0

`quantity_pad` - - - *a Quantity field of model Product*
 Additional Quantity of this Product that is consumed for each consuming actual Request. This is used to protect against mis-estimation due to poorly representative representative_configuration. Each Seller can override this with its own Product model. This Quantity is converted to the unit of this Product's `product_root`.
 Default: 0

`quote_end_of_bucket` - - - *a Logical field of model Product*
 If this field is TRUE, the first ATP quote for this Product will have as its 'dates' the `Forecast_Entry.delivery_date.end` in which the ATP exists. If this field is FALSE, the first ATP quote for this Product will have as its 'dates' the 'request.due' date if any is available.
 Default: FALSE

`quote_multiple` - - - *a Quantity field of model Product*
 ATP quotes for this Product are rounded according to this quantity. The direction of the rounding is determined by the `quote_larger_multiple` field. Additionally, if the `allocate_quote_multiple` field is TRUE, then allocations for this Product will also be rounded according to this quantity. This Quantity is converted to the unit of this Product's `product_root`.
 Default: 0

`quote_larger_multiple` - - - *a Logical field of model Product*
 If this field is TRUE, ATP quotes for this Product will be rounded up by the 'quote_multiple' for this Product. If this field is FALSE, ATP quotes will be rounded down by the 'quote_multiple' for this Product.
 Default: "true"/"yes"

`allocate_quote_multiple` - - - *a Logical field of model Product*
 If this field is TRUE, allocations for this Product will be rounded to the 'quote_multiple' for this Product. This rounding is in addition to rounding according to the 'product_root's unit which always takes place regardless of this flag. The direction (i.e. up or down) of the rounding is a function of the allocation_policy. If this field is FALSE, no additional rounding will be done for allocations of this Product.
 Default: "true"/"yes"

Model	Product Model
-------	---------------

auto_allocate_from_organization - - - *a logical field of model Product*
 If **auto_allocate_from_organization** is "true" (the default), then changes in allocation in a Forecast Entry affect the ATP of this Product at its organization. If **auto_allocate_from_organization** is "false", then changes in allocation in a Forecast Entry affect the ATP of its Generic_Products.

Currently, this field also affects where checks against ATP Entry's are performed and the ordering of consumption as Promises are made. If "true", then ATP consumption is first from the Product at its organization and the ATP Entries checked for this Product are those that are checked for this Product at its organization. If "false", then ATP consumption is first from this Product's Generic_Products and the ATP Entry's checked for this Product are those that are checked for this Product's Generic_Products.
 Default: "false"/"no"

generic_products - - - *a list of Generic_Product submodels of model Product*
 The **generic_products** are Products that represent more generic demand, demand for a class of products, or demand for products built with a certain component. It is often very difficult to forecast for specific Products, whereas it is much simpler and accurate to forecast for more generic Products. For example, it may be very difficult to forecast how many of each model of PC will sell, whereas, it may be much easier to forecast the number of 150MHz processors, the number of disk drives, and the number of keyboards.

Given that, it is preferable to setup allocations and ATP for those generic Products and allow all the different specific Products to consume ATP as needed from those generic Products. That prevents any need to prematurely allocate to specific Products, but still allows you to forecast and allocate some level of certain specific Products.

This is often used to implement a more sophisticated form of two-level (or N-level) master planning.

specific_products - - - *a list(Product) field of model Product*
 The Products which name this Product as one of its generic_products. These Products are directly allocated, but that Quantity is included in this Product's allocated Quantity. This Product is managed as a more generic Product that includes the demand for these specific_products. The specific_products are subsets of this generic Product.
 Properties: Export-Only Field

Model	Product Model
-------	---------------

active_specific_products - - - *a list(Product) field of model Product*
 Returns active Products on the specific_products list. See Product.specific_products & Product.active.
 Properties: Export-Only Field

alternate_products - - - *a list of Alternate_Product submodels of model Product*
 The alternate_products could be used as the alternates of this product. This is different from the various products of an item in the sense that the alternate products could be represented as an alternate for a generic product or as an alternate for a SKU that may have some generic products as well.

primary_products - - - *a list(Product) field of model Product*
 The Products which name this Product as one of its alternate_products. These Products are directly allocated.
 Properties: Export-Only Field

active_primary_products - - - *a list(Product) field of model Product*
 Returns active Products on the primary_products list. See Product.primary_products & Product.active.
 Properties: Export-Only Field

groups - - - *a list(Product_Group) field of model Product*
 The Product_Groups to which this Product belongs. No Product may appear in more than one Product_Group of a single Product_Group hierarchy. However, a Product may appear in any number of separate Product_Group hierarchies.

A Product that appears in no Product_Groups is considered to be its own product hierarchy, and will appear in the owner Seller's top_products.

This field cannot be edited directly; this list is determined by which Product_Groups specify this as a Sub_Product.
 Properties: Export-Only Field

top - - - *a logical field of model Product*
 If "true", then this Product appears in no Product_Groups -- it is the top of a Product hierarchy.
 This field cannot be edited directly; this list is determined by which Product_Groups specify this as a Sub_Product.
 Properties: Export-Only Field

Models	Product Model
--------	---------------

owner - - *a Seller field of model Product*

Properties: Export-Only Field

Models	Generic_Product Model
--------	-----------------------

5.1.2.3.1 Generic_Product Model

Generic_Product - - *a submodel of model Product*

The 'generic_products' are Products that represent more generic demand, demand for a class of products, or demand for products built with a certain component. It is often very difficult to forecast for specific Products, whereas it is much simpler and accurate to forecast for more generic Products. For example, it may be very difficult to forecast how many of each model of PC will sell; whereas, it may be much easier to forecast the number of 150MHz processors, the number of disk drives, and the number of keyboards.

Given that, it is preferable to setup allocations and ATP for those generic Products and allow all the different specific Products to consume ATP as needed from those generic Products. That prevents any need to prematurely allocate to specific Products, but still allows you to forecast and allocate some level of certain specific Products.

This is often used to implement a more sophisticated form of two-level (or N-level) master planning.

The Generic_Product model has fields that references these models :
Product.

These models have a field that is a Generic_Product model :
Product.

The key field for this model is product

product - - *a Product field of model Generic_Product*
The Product that represents the demand for a generic of this Product. This product's ATP can be consumed by the 'owner' Product and offered as its own (assuming all the Generic_Products have corresponding ATP available).
Default: None -- this is a key field

quantity_per - - *a Quantity field of model Generic_Product*
The Quantity of 'product' that must be consumed per unit of the 'owner' Product. This is converted to the 'product.unit'; not the 'owner.unit'.
Default: 1

owner - - *a Product field of model Generic_Product*

Properties: Export-Only Field

Model	Alternate_Product Model
-------	-------------------------

5.1.2.3.2 Alternate_Product Model

Alternate_Product -- a submodel of model Product

The 'alternate_products' could be used as the alternates of this product. This is different from the various products of an 'item' in the sense that the alternate products could be represented as an alternate for a generic product or as an alternate for a SKU that may have some generic products as well.

The Alternate_Product model has fields that references these models :
Product.

These models have a field that is a Alternate_Product model :
Product.

The key field for this model is product

product -- a Product field of model Alternate_Product
The Product that is an alternate for this Product.

Default: None -- this is a key field

quantity_per -- a Quantity field of model Alternate_Product

The Quantity of 'product' that must be consumed per unit of the 'owner' Product. This is converted to the 'product:unit'; not the 'owner:unit'.

Default: 1

rank -- a Number field of model Alternate_Product

Rank of this alternate product. This may be used in quoting and consumption to determine how to consume ATP from a group of alternates.

Default: 0

owner -- a Product field of model Alternate_Product

Properties: Export-Only Field

Model	allocation_policy submodels of model Product
-------	--

5.1.2.3.3 allocation_policy submodels of model Product

5.1.2.3.4 Product_Allocation Model

Product_Allocation -- a submodel of model Product

The 'allocations' or fixed percentages which are used by this product's owner when splitting its allocated amount up among its members. Any entry of 'allocations' whose split is set to zero is automatically deleted.

The Product_Allocation model has fields that references these models :
Seller, Product.

These models have a field that is a Product_Allocation model :
FIXED_SPLIT.

The key field for this model is member

member -- a Seller field of model Product_Allocation

A member of this Product's owner for which a fixed allocation percentage is defined.

Default: None -- this is a key field

split -- a Percentage field of model Product_Allocation

The fixed percentage split of any amount allocated to this Product's owner which should be allocated to this 'member'. The 'split' may not be less than zero.

Note that the 'split's of the 'allocations' of a Product need not add up to "100%". If

there are four 'allocations' that each have a 'split' of "100%", giving a total of "400%", the split will be even -- the same as if each had a split of "25%" for a total of "100%".

Thus, the default, "100%", will result in an even split.

Default: 100%

owner -- a Product field of model Product_Allocation

Properties: Export-Only Field

Models	Product_Group Model
--------	---------------------

5.1.2.4 Product_Group Model

Product_Group -- a submodel of model Seller

A Product_Group models a hierarchical grouping or classification of Products. A Product can only appear directly in only one Product_Group of a hierarchy. Similarly, a Product_Group can only appear directly in one place in the hierarchy. However, a Product can appear in any number of separate Product_Group hierarchies.

An implication of this is that all the Product_Groups cannot be assembled into the one (unspecified) Product_Group -- the hierarchies must remain separate, otherwise the totals for one Product would be counted dozens of times or the totals would not add in an understandable way.

For example, milk Products may be grouped by fat content: "skim", "1%", "2%", and "whole". The same milk Products may also be grouped by size: pint, quart, half-gallon, and gallon. Further, the same milk Products may be grouped by brand: "Econo-Cow" and "Pure-White". A particular Product would be in one fat content group, one size group, and one brand group.

Product_Groups are organized into hierarchies. In the above example, the four fat groups might be in the "Milk by Fat" Product_Group, the four sizes might be in the "Milk By Size" group, and the two brands might be in the "Milk By Brands" group. Further, if there are non-milk Products, those groups may be included into other Product_Groups. For instance, "Milk By Size" may be in the "By Drink By Size" Product_Group, which also includes "Orange_Juice_By_Size" and others.

Note that grouping "Milk by Fat", "Milk by Size", and "Milk by Brands" into a "Milk" group would result in counting each Milk product as many as three times. Each "Milk by *" group gives the summary of all Milk, but then breaks it out different ways.

All of the above milk Product_Groups are related to item characteristics. You can also form Product_Groups based on other properties of the Products. For instance, your Products may be divided based upon groups of customers (Site_Groups), or "market channels", or "industry segments". For example, pints of milk sold to grocery stores may be a separate Product from pints of milk sold to restaurants. A Product_Group may be formed based on that division: "Milk By Channel".

Another common division will be by delivery lead time. Often the same item will be made available as one Product with a 2-week delivery lead time and another with a 6-week delivery lead time, where the latter has a significantly lower price. When Products are so divided, they will almost always be grouped together for forecast purposes.

Models	Product_Group Model
--------	---------------------

This organization provides a great deal of flexibility in organizing and displaying information about the Products in the interactive Reports. Since Forecast accuracy improves with aggregation, being able to aggregate in many different dimensions is a key capability in forecast management. That is the purpose of Product_Group.

The Product_Group model has these submodels:

Sub_Product_Group, Sub_Product.

The Product_Group model has fields that references these models:

Product_Group, Sub_Product_Group, Unit, Seller.

These models have a field that is a Product_Group model:

Seller, Product_Group, Sub_Product_Group, Sub_Product, GROUP.

The key field for this model is name

This model may be extended with user-defined fields.

name -- a Symbol field of model Product_Group
The name of this Product_Group. This name must be unique among Product_Groups of the owner Seller.

Default: None -- this is a key field

description -- a String field of model Product_Group

A description of this Product_Group.

Default: none

top -- a Logical field of model Product_Group

If "true", then this Product_Group appears in no other Product_Groups -- it is the top of a Product_Group hierarchy.

Properties: Export-Only Field

root -- a Product_Group field of model Product_Group

The root of this Product_Group Seller tree: the corresponding Product_Group of the highest level Seller that defines a Product_Group with the same name.

Note that it is only possible to delete a Product_Group or change its name if it is the root of its Product_Group-Seller tree.

Properties: Export-Only Field

Models	Product_Group Model
--------	---------------------

`use_sid_split` - - *a Logical field of model Product_Group*
 If "true", then any addition or reduction to an forecast entry of a product_group should be distributed among its sub_groups / sub_products using the `sid_split` specified in the `Product_Group`.

If "false", then setting an forecast entry value for this product_group should distribute it to the sub_groups / sub_products such that the Percentage split among the sub_groups / sub_products remain the same.

For example, consider a `Product_Group` that contains A, B, and C, and specifies the `sid_split`s to be "60%", "30%", and "10%", respectively. Further, assume that for one Date, Range the current Forecasts are "20", "16", and "4", respectively, for a group total of "40". The group total is then set to "50". If `use_sid_split` is "false", then the current split will be followed, resulting in "25", "20", and "5", respectively. If `use_sid_split` is "true", then the additional "10" (50 - 40) will be distributed with the `sid_split`s, adding "6", "3", and "1", resulting in "26", "19", and "5", respectively.
 Default: false

`sub_groups` - - *a list of Sub_Product_Group submodels of model Product_Group*
 The `Product_Group`s that are direct members of this `Product_Group`. Note that no `Product_Group` descendant can contain this `Product_Group` or contain common Products.

`sub_products` - - *a list of Sub_Product submodels of model Product_Group*
 The Products that are direct members of this `Product_Group`. This does not include the Products in this `Product_Group`'s `sub_products`. The `all_products` field provides a complete list.

`all_products, all_products (List(Product))` - - *a List(Product) field of model Product_Group*

All of the Products that are in this `Product_Group`, both directly and indirectly. This list includes the Products in `sub_products` plus `all_products` of each of the `Product_Group`s in `sub_groups`.

If passed `List(Product)`, it returns only those Products that are in the passed List and in this `Product_Group`. In other words, it returns the intersection of the Products in this `Product_Group` and the argument List.
 Properties: Export-Only Field

Models	Product_Group Model
--------	---------------------

`all_products_and_specifics` - - *a List(Product) field of model Product_Group*
 This list includes the Products in `sub_products` plus `all_products` of each of the `Product_Group`s in `sub_groups` plus all generic_products and `specific_products` in the case of `sub_products` having related products.
 Properties: Export-Only Field

`unit` - - *a Unit field of model Product_Group*
 The unit defines the Quantity of one unit of this `Product_Group`, whether this `Product_Group` is discrete (can only exist in whole units), and what the 'preferred_measure' for this `Product_Group` is. It can also convert between different units of Measure and units of this `Product_Group`.

For example, one unit of `Product_Group X` may be "3 kg", "400 ml", and "\$12". It may be discrete, meaning that a need for "3 kg" will result in "6 kg" (2 units) being built. The 'preferred_measure' may be "kg" which will cause most fields that display a Quantity of this `Product_Group` to display in "kg" by default.

Note that this field is not settable -- you cannot assign it a new Unit. Rather, you can get the Unit of this `Product_Group` and set that Unit's fields.
 Properties: Export-Only Field

`owner` - - *a Seller field of model Product_Group*
 The Seller to which this `Product_Group` belongs.
 Properties: Export-Only Field

5.1.24.1 Sub_Product_Group Model

Sub_Product_Group -- a submodel of model Product_Group

The Product_Groups that are direct members of this Product_Group. Note that no Product_Group descendant can contain this Product_Group or contain common Products.

The Sub_Product_Group model has fields that references these models :
Product_Group, Sub_Product_Group.

These models have a field that is a Sub_Product_Group model :
Product_Group, Sub_Product_Group.

The key field for this model is product_group

product_group -- a Product_Group field of model Sub_Product_Group

A Product_Group that is a direct member of the owner' Product_Group.
Default: None -- this is a key field

quantity_per -- a Quantity field of model Sub_Product_Group

The Quantity of 'product_group' per unit of the owner' Product_Group. For example, one unit of the owner' Product_Group may be "10 gallon", "20 kg", "\$4", or "3" (three units) of this 'sub_group'. The equivalence among sub_groups must be defined in order for aggregate forecasting to be meaningful. Very often, however, the base units of all Product_Groups are the same, and thus the default of "1" (one unit to one unit) is correct. This Quantity is converted to the unit of the product_group (the "sub" group), not of the owner' Product_Group.
Default: 1

std_split -- a Percentage field of model Sub_Product_Group

The typical percentage of the owner' Product_Group's Forecast that should be allocated to this product_group. This can be used (but need not be) by GROUP Forecasts in allocating to its sub_forecasts.

Note that the std_split's of the 'sub_groups' of a Product_Group need not add up to "100%". If there are four 'sub_groups' that each have a 'std_split' of "100%", giving a total of "400%", the split will be even -- the same as if each had a split of "25%" for a total of "100%". Thus, the default, "100%", will result in an even split.
Default: 100%

root -- a Sub_Product_Group field of model Sub_Product_Group

The root of this Sub_Product_Group-Seller tree, the corresponding Sub_Product_Group of the highest level Seller that defines this 'product_group' to be a Sub_Product_Group of the corresponding 'owner' Product_Group.

Note that it is only possible to delete a Sub_Product_Group or change its 'product_group' if it is the root of its Sub_Product_Group-Seller tree.

Properties: Export-Only Field

owner -- a Product_Group field of model Sub_Product_Group

Properties: Export-Only Field

Model	Sub_Product Model
-------	-------------------

5.1.2.4.2 Sub_Product Model

Sub_Product -- a submodel of model Product_Group

The Products that are direct members of this Product_Group. This does not include the Products in this Product_Group's sub_products'. The all_products' field provides a complete list.

The Sub_Product model has fields that references these models :
Product, Sub_Product, Product_Group.

These models have a field that is a Sub_Product model :
Product_Group, Sub_Product.

The key field for this model is product

product -- a Product field of model Sub_Product
A Product that is a direct member of the owner' Product_Group.
Default: None -- this is a key field

quantity_per -- a Quantity field of model Sub_Product
The Quantity of product per unit of this Product_Group. For example, one unit of the owner' Product_Group may be "10 gallon", "20 kg", "\$4", or "3" (three units) of this product'. The equivalence among sub_products and the owner' Product_Group must be defined in order for aggregate forecasting to be meaningful. Very often, however, the base units of all Product_Groups are the same, and thus the default of "1" (one unit to one unit) is correct. This Quantity is converted to the unit' of the product' (the "sub" product), not of the owner' Product_Group.
Default: 1

std_split -- a Percentage field of model Sub_Product
The typical percentage of the owner' Product_Group's Forecast that should be allocated to this product'. This can be used (but need not be) by GROUP Forecasts in allocating to its sub_forecasts'. Note that if the splits add up to more than 100%, they will be normalized to 100% when used to split the forecast.
Default: 1

root -- a Sub_Product field of model Sub_Product
The root of this Sub_Product-Seller tree; the corresponding Sub_Product of the highest level Seller that defines this product' to be a Sub_Product of the corresponding owner' Product_Group.

Model	Sub_Product Model
-------	-------------------

Note that it is only possible to delete a Sub_Product or change its product' if it is the root of its Sub_Product-Seller tree.
Properties: Export-Only Field

owner -- a Product_Group field of model Sub_Product
Properties: Export-Only Field

Models	Plan Model
--------	------------

5.1.3 Plan Model

Plan -- a submodel of model Supply_Chain

The Plan models what is happening and what is planned to be done in a Supply_Chain. The Supply_Chain defines what is possible; the Plan defines the current plans for using the Site capabilities.

Plan's structure mirrors that of Supply_Chain. As Supply_Chain is made up of Sites, Plan is made up of Site_Plans. As Site is made up of Operations, Resources, and Buffers, Site_Plan is made up of Operation_Plans, Resource_Plans, and Buffer_Plans. Supply_Chain contains the "master" data; Site contains the "live" data.

Plans define an Edit-Boundary: edit-security and edit-update notification for all models under Plan are controllable at the Plan level.

This split is useful for divisions to support What-If analysis. You may want to consider a particular Supply_Chain in different states with different plans. For example, what if an oven Resource goes down? What if we receive triple the demand we have forecasted? What if we buy a new machine? What if we run these jobs earlier? What if these arrive late?

The static specification mechanisms of the Plan models are designed to support the "minimal transaction" philosophies, such as Just-In-Time (JIT). As time marches forward, we simply assume everything is going according to the Plan, unless told otherwise in the form of state "overrides".

The Plan model has these submodels:
Site_Plan, Seller_Plan, Problem, Active_Strategy.

The Plan model has fields that references these models:
Site_Plan, Seller_Plan, Problem, Active_Strategy, Resource_Plan, Buffer_Plan, Forecast, Supply_Chain.

These models have a field that is a Plan model:
Supply_Chain, Site_Plan, Seller_Plan, Problem, Active_Strategy.

The key field for this model is name
This model may be extended with user-defined fields.

name -- a Symbol/field of model Plan
The name of this Plan.

Models	Plan Model
--------	------------

Default: None -- this is a key field

description -- a String/field of model Plan
A description of this Plan.

Default: none

current -- a Date/field of model Plan

The Date for which this Plan is being computed; the Date this Plan is expected to be put into place. Plans before this Date should generally not be adjusted since it will be too late by the time this Plan is put into place.

If the current planning effort is trying to prepare a Plan that will go into effect tomorrow morning, then Plan::current should be set to tomorrow morning such that the plans between now and tomorrow morning are treated as if they have already occurred, and therefore are not open for replanning.

If the 'current' Date is set beyond the 'horizon', the 'horizon' will be expanded to enclose 'current'.

Note that this Date moves independent of 'now' and 'horizon'. The 'current' Date will move with the most detailed planning cycle, whereas 'horizon' will move with the least detailed. For example,

Note that there is no undo from setting 'current'. For this reason, it is a good idea to save your plan prior to setting 'current'.
Default: 'now'

horizon -- a Date/Range/field of model Plan
The planning horizon of this Plan. The Date_Range over which this Plan spans.

The 'horizon.end', which must be after 'current', defines the amount of time the Plan extends into the future. The 'horizon.start', which must be at or before 'current', defines the amount of time the Plan extends into the past.

Note that these dates move independent of 'now' and 'current'. Generally, these Dates are moved in conjunction with a regular planning cycle. For example, the 'horizon' will be moved forward each month in conjunction with importing the new forecast information which is computed monthly. In contrast, 'current' may move forward daily our hourly during that monthly planning cycle, as detailed adjustments are made. Traditional software that does not support multiple granularities often does not need a horizon into the past since 'current' is as far back as it needs to go.

Model	Plan Model
-------	------------

Note that there is no undo from setting 'horizon'. For this reason, it is a good idea to save your plan prior to setting 'horizon'.

Default: from the start of this month until a year from then

default_operation_plan_rank - - a Expression *field of model Plan*

An Expression that is passed an Operation_Plan as 'it' and is executed whenever a new top Operation_Plan is created. Every top Operation_Plan created will get its rank assigned upon creation in this manner.

Default: 0

site_plans - - a list of Site_Plan *submodels of model Plan*

The plans for each Site defined by the owner Supply_Chain. Each site will have a Site_Plan per Plan in the owner Supply_Chain. So, if the Supply_Chain has two plans (one production plan and the other what-if plan), each site will have two Site_Plans, one each per Plan.

top_site_plans - - a List(Site_Plan) *field of model Plan*

The plans for each Site in the owner Supply_Chain's list of 'top_sites'. These Sites are the topmost organizations; they are not within another organization. This list is particularly useful as a starting point for displaying the hierarchy of Site_Plans.

Properties: Export-Only Field

seller_plans - - a list of Seller_Plan *submodels of model Plan*

The plans for the sales personnel and sales organizations responsible for forecasting and selling this supply_chain's Products. It includes those Forecasts, the Requests generated by those Forecasts, and customer Requests through those Sellers.

top_seller_plans - - a List(Seller_Plan) *field of model Plan*

The plans for each Seller in the owner Supply_Chain's list of 'top_sellers'. These Sellers are the topmost organizations; they are not within another organization. This list is particularly useful as a starting point for displaying the hierarchy of Seller_Plans.

Properties: Export-Only Field

the_problems - - a list of Problem *submodels of model Plan*

The Problems that have been detected in this Plan. These may be feasibility Problems, or simply undesirable situations (such as extra cost incurred to run overtime).

problems, **problems (Date_Range)**, **problems (Problem_Category)**, **problems (Date_Range, Problem_Category)** - - a List(Problem) *field of model Plan*

The Problems detected with this Plan. If passed a Date_Range, only the Problems whose 'dates' overlap are returned. If passed a Problem_Category, only the Problems with that category are returned.

Model	Plan Model
-------	------------

Note that you can pass in one of the special Problem_Category's to get Problems in larger groups. For example, the OPERATION_Problem_Category will give all Problems in Operation-related Problem categories. Similarly for RESOURCE_BUFFER, and even ALL.

Properties: Export-Only Field

problem_categories - - a List(Problem_Category) *field of model Plan*

For each different category of Problem in 'problems', a Problem_Category is added to this list. It gives you the name of the 'category' (in various forms) plus a list of just the Problems of that 'category'. These sublists are often much easier to deal with than the full 'problems' list.

Properties: Export-Only Field

top_active_strategies - - a List(Active_Strategy) *field of model Plan*

A list of the top level Active_Strategies in the plan.

Properties: Export-Only Field

active_strategies - - a list of Active_Strategy *submodels of model Plan*

Each Problem_Set defines a set of Problems to be addressed by this Active_Strategy. The Problems are specified by category and tolerances within a certain horizon. The relative 'focus' that the Active_Strategy will place on each Problem is also specified.

auto_run_strategy - - a Active_Strategy *field of model Plan*

The Active_Strategy that is specified with 'auto_run'='true'. It is run automatically after each plan change.

Properties: Export-Only Field

background_run_strategy - - a Active_Strategy *field of model Plan*

The Active_Strategy that is specified with 'background_run'='true', if there is one. It is run automatically when the planning engine is otherwise idle.

Properties: Export-Only Field

resource_plan (Resource) - - a Resource_Plan *field of model Plan*

Returns the Resource_Plan of this Plan for the specified Resource.

Note that 'plan_resource_plan(resource)' is equivalent to

'plan_site_plans.find(resource.owner).resource_plans.find(resource)'.

Properties: Export-Only Field

Models	Plan Model
--------	------------

buffer_plan (Buffer) -- *a Buffer_Plan field of model Plan*
Returns the Buffer_Plan of this Plan for the specified Buffer.

Note that **plan.buffer_plan(buffer)** is equivalent to **plan.site_plans.find(buffer.owner).buffer_plans.find(buffer)**.
Properties: Export-Only Field

operation_plans (Operation) -- *a List(Operation_Plan) field of model Plan*
Returns the Operation_Plans of this Plan for the specified Operation.

Note that **plan.operation_plans(operation)** is equivalent to **plan.site_plans.find(operation.owner).operation_plans.filter(#operation == operation)**.
Properties: Export-Only Field

forecast (Product) -- *a Forecast field of model Plan*
Returns the Forecast of this Plan for the specified Product.

Note that **plan.forecast(product)** is equivalent to **plan.seller_plans.find(product.owner).forecast.find(product)**.
Properties: Export-Only Field

run_for_now (Integer) -- *a Void field of model Plan*
Temporary run field for now.
Properties: command=True Export-Only Field

owner -- *a Supply_Chain field of model Plan*
Properties: Export-Only Field

Models	Site_Plan Model
--------	-----------------

5.1.3.1 Site_Plan Model

Site_Plan -- *a submodel of model Plan*

The **Site_Plan** models the plans for a **Site**. As **Sites** can be made up of **Operations**, **Resources**, and **Buffers**, so can **Site_Plans** can be made up of **Operation_Plans**, **Resource_Plans**, and **Buffer_Plans**.

The model has selectors:
role.

The **Site_Plan** model has fields that references these models :
Site, **Site_Plan**.

These models have a field that is a **Site_Plan** model :
Site, **Plan**, **Operation_Plan**, **Resource_Plan**, **Buffer_Plan**, **Site_Plan**, **Operation_State**, **Request**, **Promise**, **Acceptance**, **Delivery_Request**.

The key field for this model is **site**
This model may be extended with user-defined fields.

site -- *a Site field of model Site_Plan*
The **Site** for which this plan pertains.
Default: None -- this is a key field

description -- *a String field of model Site_Plan*
A description of this **Site_Plan**.
Default: none

organization_plan -- *a Site_Plan field of model Site_Plan*
The **Site_Plan** for this site's 'organization'.
Properties: Export-Only Field

members -- *a List(Site_Plan) field of model Site_Plan*
The **Site_Plans** for **Sites** which specify this **Site** as their 'organization'.
Properties: Export-Only Field

role -- *an extension selector of model Site_Plan*
This is the same value as the **site.role**. Here, it adds to the **Site_Plan** the fields associated with the **Site**'s role in the **Supply_Chain**. For instance, **CUSTOMER** **Sites** do not have 'requests' and 'promises', but **LINK** and **SUPPLIER** **Sites** do.

Models	Site_Plan Model
--------	-----------------

Properties: extension_name_as_model=Site Export-Only Field
 Extensions: LINK, SUPPLIER, CUSTOMER.

owner - - a Plan field of model Site_Plan

Properties: Export-Only Field

Models	Standard Extensions of model Site_Plan
--------	--

5.1.3.1.1 Standard Extensions of model Site_Plan
 5.1.3.1.1.1 role extensions of model Site_Plan
 5.1.3.1.1.1 LINK -- a role extension of model Site_Plan

A LINK Site is considered to be a "link" of this supply "chain".

The LINK model has these submodels :
 Buffer_Plan, Resource_Plan, Operation_Plan, Operation_State, Request, Promise, Acceptance.

The LINK model has fields that references these models :
 Buffer_Plan, Resource_Plan, Operation_Plan, Operation_State, Request, Promise, Acceptance.

buffer_plans - - a list of Buffer_Plan submodels of model LINK

The plans for each Buffer defined in the site_plan site. Each buffer in each Site will have exactly one Buffer_Plan per Site_Plan.

resource_plans - - a list of Resource_Plan submodels of model LINK

The plans for each Resource defined in this site_plan's site. Each resource in each Site will have one Resource_Plan per Site_Plan.

operation_plans - - a list of Operation_Plan submodels of model LINK

The plans for each Operation defined in this site_plan's site. Each operation in each Site could have multiple Operation_Plan per Site_Plan.

operation_states - - a list of Operation_State submodels of model LINK

Static reports which will be assigned to Operation_Plan.

requests - - a list of Request submodels of model LINK

The Requests that have been made by other Sites for Items supplied by this Site. This is the raw demand on this Site.

promises - - a list of Promise submodels of model LINK

The promises that have been made by this Site to supply Items to other Sites. This is the demand this Site has agreed to fulfill. By some definitions, this is the master production schedule.

acceptances - - a list of Acceptance submodels of model LINK

The acceptances that have been made by the other Site in response to the promises from this site

Models	LINK Extension
--------	----------------

plan_to_satisfy_unanswered_requests - - - *a Void field of model LINK*
This command creates plans to satisfy all the Requests that have been made upon this Site_Plan that have not yet been "answered". It does this by simply calling plan_to_satisfy on each of its "unanswered" requests.

Alternatively, plan_to_satisfy_all_promises can create plans such that all Promises would be satisfied; or plan_to_satisfy_all_requests to create plans such that all Requests upon this Site_Plan are satisfied.
Properties: command=True Export-Only Field

plan_to_satisfy_queued_requests - - - *a Void field of model LINK*
This command creates plans to satisfy all the Requests that have been made upon this Site_Plan that have been "queued" for later consideration. It does this by simply calling plan_to_satisfy on each of its "queued" requests.

Alternatively, plan_to_satisfy_all_promises can create plans such that all Promises would be satisfied; or plan_to_satisfy_all_requests to create plans such that all Requests upon this Site_Plan are satisfied.
Properties: command=True Export-Only Field

plan_to_satisfy_all_requests - - - *a Void field of model LINK*
This command creates plans to satisfy all the Requests that have been made upon this Site_Plan. It does this by simply calling plan_to_satisfy on each of its requests. Thus, it is equivalent to requests_for_each(plan_to_satisfy).

Alternatively, plan_to_satisfy_all_promises can create plans such that all Promises would be satisfied; or plan_to_satisfy_unanswered_requests to create plans such that Requests that have not yet been "answered" are satisfied.
Properties: command=True Export-Only Field

plan_to_satisfy_all_promises - - - *a Void field of model LINK*
This command creates plans to satisfy all the Promises that have been made by this Site_Plan. It does this by simply calling plan_to_satisfy on each of its promises. Thus, it is equivalent to promises_for_each(plan_to_satisfy).

Alternatively, plan_to_satisfy_all_requests can create plans such that all Requests would be satisfied; or plan_to_satisfy_unanswered_requests to create plans such that Requests that have not yet been "answered" are satisfied.
Properties: command=True Export-Only Field

Models	LINK Extension
--------	----------------

promise_as_planned - - - *a Void field of model LINK*
This command sets this Site_Plan's promises to promise what has been planned for each. It does this by simply calling promise_as_planned on each of its promises. Thus, it is equivalent to promises_for_each(promise_as_planned).
Properties: command=True Export-Only Field

supply_requests - - - *a List(Request) field of model LINK*
The Requests that have been made by this Site for Items supplied by other Sites.
Properties: Export-Only Field

supply_promises - - - *a List(Promise) field of model LINK*
The Promises that have been made by other Sites to supply Items to this Site.
Properties: Export-Only Field

supply_item_requests, **supply_item_requests(Date, Range)**,
supply_item_requests(item), **supply_item_requests(item, Date, Range)** - - - *a List(Item, Request) field of model LINK*
The Requests that have been made by this Site for Items supplied by other Sites.
Properties: Export-Only Field

item_demand (List(Item), Date, Range) - - - *a Quantity field of model LINK*
Returns the sum of demand for a List of Items during a Date_Range. The Quantity will be unitless as the sum of the number of units of each Item.
Properties: Export-Only Field

problems, **problems(Date, Range)**, **problems(Problem, Category)**, **problems(Problem, Category, Date, Range)** - - - *a List(Problem) field of model LINK*
The Problems detected with this Site_Plan. If passed a Problem_Category, only the Problems with that category are returned. If passed a Date_Range, only the Problems whose dates overlap are returned.

Note that you can pass in one of the special Problem_Category's to get Problems in larger groups. For example, the OPERATION Problem_Category will give all Problems in Operation-related Problem categories. Similarly for RESOURCE, BUFFER, and even ALL.
Properties: Export-Only Field

Model	SUPPLIER Extension
-------	--------------------

problem, categories - - a List(Problem_Category) field of model LINK
For each different 'category' of Problem in 'problems', a Problem_Category is added to this list. It gives you the name of the 'category' (in various forms) plus a list of just the Problems of that 'category'. These sublists are often much easier to deal with than the full 'problems' list.

Note that this List will not include special Problem_Categories, such as OPERATION, RESOURCE, BUFFER, or ALL.

Properties: Export-Only Field

5.1.3.1.1.2

SUPPLIER -- a role extension of model Site_Plan

A SUPPLIER Site is not planned or modeled in detail; rather it represents the items that can be procured from a supplier by LINK Sites, the Requests for those items, and the Promises received from the supplier.

The SUPPLIER model has these submodels :
Request, Promise, Acceptance.

The SUPPLIER model has fields that references these models :
Request, Promise, Acceptance.

requests - - a list of Request submodels of model SUPPLIER
The Requests that have been made by other Sites for items supplied by this Site. This is the raw demand on this Site.

promises - - a list of Promise submodels of model SUPPLIER
The promises that have been made by this Site to supply items to other Sites. This is the demand this Site has agreed to fulfill.

acceptances - - a list of Acceptance submodels of model SUPPLIER
The acceptances that have been made by the other Site in response to the promises from this site

5.1.3.1.1.3

CUSTOMER -- a role extension of model Site_Plan

A CUSTOMER Site is not planned or modeled in detail; rather, it is simply a destination for deliveries and source of Requests.

Model	role submodels of model Site_Plan
-------	-----------------------------------

5.1.3.1.2 role submodels of model Site_Plan

5.1.3.1.3 Buffer_Plan Model

Buffer_Plan -- a submodel of model Site_Plan

A Buffer is a model of the management of an item at a particular Location (a SKU). Most material/inventory planning functionality is handled by Buffer.

The model has selectors:

flow_policy.

The Buffer_Plan model has these submodels :

Lot.

The Buffer_Plan model has fields that references these models :
Buffer, Lot, Operation_Plan, Site_Plan.

These models have a field that is a Buffer_Plan model :
Plan, Buffer, LINK, Flow_Plan, Lot, PRODUCE, CONSUME,
NEGATIVE_ON_HAND, OVER_FLOW_LIMIT,
NEGATIVE_ON_HAND_AT_END, LOT_OVER_CONSUMED,
LOT_NOT_CONSUMED, LOT_NOT_PRODUCED,
LOT_OVER_PRODUCED, LOW_ON_HAND, EXCESS_ON_HAND,
EXCESS_ON_HAND_AT_END, BUFFER_CHARGES, Sorted_Bucket.

The key field for this model is buffer
This model may be extended with user-defined fields.

buffer - - a Buffer field of model Buffer_Plan

The Buffer that this is planning.

Default: None -- this is a key field

remark - - a String field of model Buffer_Plan
Comments about this specific plan.
Default: none

flow_plans, flow_plans(Date_Range), flow_plans(Date_Range, Logical) - - a List(Flow_Plan) field of model Buffer_Plan
The Flows planned to produce into or consume from this Buffer.
Properties: Export-Only Field

Models	Buffer_Plan Model
--------	-------------------

producing_flow_plans, producing_flow_plans (Date, Range) ,
producing_flow_plans (Date, Range, Logical) -- a List(Flow_Plan) field of model
Buffer_Plan
 The Flows planned to produce into this Buffer.
 Properties: Export-Only Field

consuming_flow_plans, consuming_flow_plans (Date, Range) ,
consuming_flow_plans (Date, Range, Logical) -- a List(Flow_Plan) field of
model Buffer_Plan
 The Flows planned to consume from this Buffer.
 Properties: Export-Only Field

producing_flow (Date, Range) -- a Quantity field of model Buffer_Plan
 The total Quantity of Flow_Plan producing into this Buffer during the given
 Date_Range.
 Properties: Export-Only Field

consuming_flow (Date, Range) -- a Quantity field of model Buffer_Plan
 The total Quantity of Flow_Plan consuming from this Buffer during the given
 Date_Range. Note that this will be a negative number.
 Properties: Export-Only Field

on_hand_profile, on_hand_profile (Date, Range) -- a List(Profile_Quantity)
field of model Buffer_Plan
 A List of the changes to the on_hand Quantity profile during the specified finite
 Date_Range.
 Properties: command=True Export-Only Field

on_hand, on_hand (Date) , on_hand (Date, Range) , on_hand (Date, Range,
Logical) , on_hand (Flow_Plan) -- a Quantity field of model Buffer_Plan
 If a Date is passed, this returns the Quantity planned to be on-hand as of that Date. If a
 Period is passed, this returns the smallest Quantity planned to be on-hand during that
 Period. If a Period and a Logical are passed, this returns the smallest Quantity planned
 to be on-hand during the Period if the Logical is True. If it is False, this returns the
 largest Quantity planned during the Period. If a Flow_Plan is passed, this returns the
 Quantity planned to be on-hand at the completion of the Flow_Plan. If the Date_Range
 is passed with the logical argument's value equal to TRUE, it will return smallest

Models	Buffer_Plan Model
--------	-------------------

Quantity planned to be on-hand during that date range. If the logical argument's value
 is FALSE, it will return the max Quantity planned to be on-hand during that date
 range. If no argument is passed, this returns the Quantity on-hand on 'on_hand_date'. It
 returns 'nonexistent' if the argument Date is earlier than 'on_hand_date'. The Quantity
 is converted to the 'unit' of this buffer.

Setting the field sets 'on_hand_date' to the argument Date, or to 'now' if no argument,
 and sets this field to report the specified Quantity as on_hand.

Be careful when setting this to make sure that it is set consistent with the state infor-
 mation of the Operation_Plan. For example, suppose an Operation_Plan plans to
 remove 500 units at 11:00, but in reality the units are removed at 10:00. If the drop in
 'on_hand' is reported (this is set to 500 less), but the units are not reported to have
 'arrived' at the Operation_Plan, then it will appear that at 11:00 an additional 500 units
 will be removed. Note also that reporting the arrival at the Operation_Plan will auto-
 matically remove the Quantity from on_hand.
 Default: 0

Properties: Export-Only Field
on_hand_date -- a Date field of model Buffer_Plan
 The Date at which 'on_hand' was reported to be accurate.
 Default: infinite_past

on_handLots (Date) -- a List(Lot_Flow) field of model Buffer_Plan
 A list of all Lot_Flows that make up the on hand amount on the date given.
 Properties: Export-Only Field

lots -- a list of Lot submodels of model Buffer_Plan
 The specific lots planned to flow through this in this Buffer. Note that lots are not
 tracked for STANDARD items.

flow_policy -- an extension selector of model Buffer_Plan
 This has the same value as 'buffer_flow_policy'. This extension defines how the Opera-
 tion Flows placed on this Buffer are planned. The associated fields define the Problems
 that can be caused by the flow of items through this Buffer, and how those Problems
 are resolved. In particular, it defines the relationship between the flows consuming
 from the Buffer and those producing into it. For example, Lot-For-Lot, Fixed, EOQ,
 and POQ inventory policies would be implemented as different flow_policy's.
 Properties: extension_same_as_model=Buffer Export-Only Field
 Extensions:

HUCKETED_NESTED_SORT, PRODUCING_FLOW_CALENDAR,
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK,
ON_HAND_CALENDAR, ON_HAND_CALENDAR_FILTER_AND_RANK,
FLOW_LIMIT_CALENDAR,
FLOW_LIMIT_CALENDAR_FILTER_AND_RANK, BASIC,
BASIC_FILTER_AND_RANK, FIXED_QUANTITY_MULTIPLE,
MULTIPLE_FILTER_AND_RANK, FIXED_QUANTITY_FENCED,
FIXED_TIME.

create_producing_operation_plan (Measure_Unit, Restriction) -- a

Operation_Plan field of model Buffer_Plan

If this Buffer has a producing_operation designated, then

create_producing_operation_plan will create a new Operation_Plan of the designated
Quantity and Restriction, motivated to PRODUCE into this Buffer_Plan.

Properties: command=True Export-Only Field

create_consuming_operation_plan (Measure_Unit, Restriction) -- a

Operation_Plan field of model Buffer_Plan

If this Buffer has a consuming_operation designated, then

create_consuming_operation_plan will create a new Operation_Plan of the designated
Quantity and Restriction, motivated to CONSUME from this Buffer_Plan.

Properties: command=True Export-Only Field

problems, problems (Date_Range) , problems (Problem_Category) , problems

(Date_Range, Problem_Category) -- a List(Problem) field of model Buffer_Plan

The Problems detected with this Buffer_Plan. If passed a Date_Range, only the Prob-
lems whose 'dates' overlap are returned. If passed a Problem_Category, only the Prob-
lems with that 'category' are returned.

Properties: Export-Only Field

problem_categories -- a List(Problem_Category) field of model Buffer_Plan

For each different 'category' of Problem in 'problems', a Problem_Category is added to
this list. It gives you the name of the 'category' (in various forms) plus a list of just the
Problems of that 'category'. These sublists are often much easier to deal with than the
full 'problems' list.

Properties: Export-Only Field

in_flow_plans, in_flow_plans (Date_Range) , in_flow_plans (Date_Range, Logi-
cal) -- a List(Flow_Plan) field of model Buffer_Plan

Obsolete! This field has been renamed producing_flow_plans in an effort to have
more consistent and less ambiguous naming. This field will be eliminated in a future
release.

Properties: obsolete=True Export-Only Field

out_flow_plans, out_flow_plans (Date_Range) , out_flow_plans (Date_Range,
Logical) -- a List(Flow_Plan) field of model Buffer_Plan

Obsolete! This field has been renamed consuming_flow_plans in an effort to have
more consistent and less ambiguous naming. This field will be eliminated in a future
release.

Properties: obsolete=True Export-Only Field

in_flow (Date_Range) -- a Quantity field of model Buffer_Plan

Obsolete! This field has been renamed producing_flow in an effort to have more con-
sistent and less ambiguous naming. This field will be eliminated in a future release.

Properties: obsolete=True Export-Only Field

out_flow (Date_Range) -- a Quantity field of model Buffer_Plan

Obsolete! This field has been renamed consuming_flow in an effort to have more con-
sistent and less ambiguous naming. This field will be eliminated in a future release.

Note that this will be a negative number.

Properties: obsolete=True Export-Only Field

owner -- a Slic_Plan field of model Buffer_Plan

The Slic_Plan to which this Buffer_Plan belongs.

Properties: Export-Only Field

Models	Lot Model
--------	-----------

5.1.3.1.3.1 Lot Model

Lot -- a submodel of model Buffer_Plan

A Lot model's a quantity of an Item that all has common characteristics. In some industries these may be called "batches", "loads", "rolls", "coils", "ingots", "melts", "dye lots", etc. Note that lots are not tracked for STANDARD items.

The Lot model has fields that references these models :

Configuration, Lot_Flow, Buffer_Plan.

These models have a field that is a Lot model :

Buffer_Plan, Lot_Flow.

The key field for this model is name

This model may be extended with user-defined fields.

name -- a Symbol field of model Lot

The name of this Lot. If not specified, then it is derived from the producing_flow.flow_plan. The operation_plan.release_name' is concatenated a hyphen and the flow name. Note that most Lots will not be given unique names until they come into existence on the floor. Sometimes, not until they have reached a Buffer.

Default: None -- this is a key field

remark -- a String field of model Lot

Remarks about this particular Lot.

Default: none

quantity (Date) -- a Quantity field of model Lot

The Quantity of configuration' that is in this Lot. This Quantity is converted to the unit of the owner.buffer.

Default: 0

configuration -- a Configuration field of model Lot

The lot-specific information recorded for the items of this Lot. What information is provided is specified by the 'spec' extension of the 'item', where the 'item' will be the same as owner.buffer.item.
Default: [unspecified]

Models	Lot Model
--------	-----------

formed -- a Logical field of model Lot

If "true", then this Lot has been formed -- there are 'quantity' of 'configuration' in existence. If "false", then this Lot is not yet formed -- it may be planned to exist, or otherwise just under consideration.

Default: false

producing_flow -- a Lot_Flow field of model Lot

The Lot_Flow (and Flow_Plan) that produces this Lot into this Buffer.

Properties: Export-Only Field

consuming_flows -- a List(Lot_Flow) field of model Lot

The Lot_Flows (and Flow_Plans) that consume this Lot from this Buffer.

Properties: Export-Only Field

supplying_flow -- a Lot_Flow field of model Lot

Obsolete! This field has been renamed 'producing_flow' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.

Properties: obsolete=True Export-Only Field

owner -- a Buffer_Plan field of model Lot

The Buffer_Plan through which this Lot flows.

Properties: Export-Only Field

Models	flow_policy submodels of model Buffer_Plan
--------	--

5.1.3.1.3.2 flow_policy submodels of model Buffer_Plan

5.1.3.1.3.3 Sorted_Bucket Model

Sorted_Bucket -- a submodel of model Buffer_Plan

The buckets that are created by the Buffer's horizon' for this Buffer_Plan.

The Sorted_Bucket model has fields that references these models :
Buffer_Plan.

These models have a field that is a Sorted_Bucket model :
BUCKETED_NESTED_SORT, BUCKETED_COMBINED_SORT.

The key field for this model is dates
This model may be extended with user-defined fields.

dates -- a Date_Range field of model Sorted_Bucket
The Date_Range covered by this bucket, as defined by the Buffer's horizon' ;
Default: None -- this is a key field

ending_on_hand -- a Quantity field of model Sorted_Bucket
The on_hand that should be present at the end of this bucket (dates.end) as computed from the sum of the Buffer fixed_ending_on_hand and the per_next_ending_on_hand multiplied by the consuming_flow in the next bucket. A Problem (EXCESS_ON_HAND, LOW_ON_HAND, or NEGATIVE_ON_HAND) will be created if the on_hand at the end of the bucket is not equal to this 'ending_on_hand' Quantity.

This field, though typically computed, is settable. Setting 'ending_on_hand' will also set 'lock_ending_on_hand' to "true" -- which will prevent this field from being recomputed from the Buffer fields -- it will remain the value set into it.

This allows the user to easily manipulate the target ending_on_hand value for a particular month, without needing to modify the Buffer parameters. This is often used to "release" some of the planned inventory in the near-term if supply is not sufficient to cover demand.

Default: computed from the Buffer fields

Models	Resource_Plan Model
--------	---------------------

lock_ending_on_hand -- a Logical field of model Sorted_Bucket
If "false", then the 'ending_on_hand' is computed from the sum of the Buffer fixed_ending_on_hand and the per_next_ending_on_hand multiplied by the consuming_flow in the next bucket. If "true", then the current 'ending_on_hand' setting will be used, overriding the setting that would normally be computed based upon the fields in the Buffer.

This allows the user to easily manipulate the target ending_on_hand value for a particular month, without needing to modify the Buffer parameters. This is often used to "release" some of the planned inventory in the near-term if supply is not sufficient to cover demand.
Default: false

producing_flow_plans -- a List(Flow_Plan) field of model Sorted_Bucket
The Flow_Plans that produce into this Buffer_Plan during this Sort_Bucket.

If these Flow_Plan were motivated by this Buffer_Plan, then they will correspond to the specified criteria that are marked 'produce_separately'.
Properties: Export-Only Field

consuming_flow_plans -- a List(Flow_Plan) field of model Sorted_Bucket
The Flow_Plans that consume from this Buffer_Plan during this Sort_Bucket, sorted according to the Buffer 'flow_policy' and the specified sort criteria.
Properties: Export-Only Field

owner -- a Buffer_Plan field of model Sorted_Bucket

Properties: Export-Only Field

5.1.3.1.4 Resource_Plan Model

Resource_Plan -- a submodel of model Site_Plan

A Resource models the machines, tools, fixtures, labor, and other things that are used by Operations in transforming items. The capacity of a Plan to perform Operations is modeled by the Resources.

Resource_State models the "current" state of a Resource: its current setup, its current location, its current operating efficiency (0% means its down), and its current activity.

The model has selectors:
efficiency, load_policy, size, maintenance.

The Resource_Plan model has fields that references these models :
Resource, Location, Skill, Site_Plan.

These models have a field that is a Resource_Plan model :

Plan, Resource, LINK, Load_Plan, TRANSIT, SETUP, SKILL, MAINTENANCE, Consolidation, UNCONSOLIDATED, UNCOORDINATED, CONSOLIDATION_OVERSIZE, OVERLOAD, OVERTIME, OVERSIZE, BUCKET_OVERSIZE, UNDERLOAD, CUMULATIVE_OVERLOAD.

The key field for this model is resource
This model may be extended with user-defined fields.

resource -- *a Resource field of model Resource_Plan*
The Resource for which this Resource_Plan is planning.
Default: None -- this is a key field

remark -- *a String field of model Resource_Plan*
Comments about this specific plan.
Default: none

efficiency_profile (Date_Range) -- *a List(Profile_Percentage) field of model*

Resource_Plan
A List of all changes in the efficiency profile during the specified finite Date_Range.
This is the resultant efficiency profile generated by combining efficiency specified by resource efficiency extension and all the efficiency changes made during planning by incurring extra cost. Note, that this is just the resource efficiency ignoring the efficiency at particular skill.

Properties: Export-Only Field

efficiency_average (Date_Range) -- *a Percentage field of model Resource_Plan*
The average efficiency during a particular Date_Range (or at a particular Date).
Properties: Export-Only Field

efficiency_profile_at_skill (Skill, Date_Range) -- *a List(Profile_Percentage) field of model Resource_Plan*

A List of all changes in the efficiency profile during the specified finite Date_Range.
Values of this profile combines efficiencies specified by resource efficiency extension, plus the efficiency changes made during planning, by incurring extra cost and efficiency of 'resource' at performing specified skill. If the skill passed in as an argument is unspecified, then skill efficiency will be ignored. If the resource does not belong to the skill group specified by the skill, it will return 0 efficiency.

Properties: Export-Only Field

efficiency_average_at_skill (Skill, Date_Range) -- *a Percentage field of model Resource_Plan*

The average efficiency during a particular Date_Range (or at a particular Date). Average efficiency value of this profile combines efficiencies specified by resource efficiency extension, plus the efficiency changes made during planning, by incurring extra cost and efficiency of resource at performing specified skill.

Properties: Export-Only Field

available_time (Date_Range) -- *a Time field of model Resource_Plan*

The total actual hours that this Resource is planned to be available (efficiency > 0%) during a particular Date_Range. These are actual hours, not efficiency-adjusted standard hours. See the capacity efficiency-adjusted standard hours.

Properties: Export-Only Field

capacity (Date_Range) -- *a Time field of model Resource_Plan*

The total efficiency-adjusted standard hours of capacity available from this Resource during a particular Date_Range. This is the standard hours of load that can be handled by this Resource during the given dates (see load_std_time).

Mathematically, 'capacity(pd)' is the same as 'pd.end - pd.start' * efficiency_average(pd), which is equivalent to the traditional definition of capacity, the available_time * the efficiency during that time. For actual hours of availability, see 'available_time'.

Properties: Export-Only Field

efficiency -- *an extension selector of model Resource_Plan*
This has the same value as 'resource.efficiency'. This extension defines how the availability, capacity, and efficiency of this Resource is modeled. The associated fields describe the efficiency of this Resource over time.

Properties: extension_same_as_model=Resource Export-Only Field

load_plans, load_plans (Date_Range) -- a List[Load_Plan] field of model Resource_Plan

The Loads that are planned on this Resource, excluding self-imposed setup and maintenance. If passed a Date_Range, only the Load_Plan's that overlap that Date_Range are returned.

Properties: Export-Only Field

load_time (Date_Range) -- a Time field of model Resource_Plan

The total actual hours of load that is planned on this Resource during a particular Date_Range. This excludes self-imposed setup, change and maintenance time. For comparison, 'load_time(gd)' is comparable to 'available_time(gd)'. See 'load_sid_time' for total 'standard hours' of load, which is comparable to 'capacity'.

Properties: Export-Only Field

load_sid_time (Date_Range) -- a Time field of model Resource_Plan

The total "standard hours" of load that is planned on this Resource during a particular Date_Range. This excludes self-imposed setup, change and maintenance time. For comparison, 'load_sid_time(gd)' is comparable to 'capacity(gd)'. See 'load_time' for total actual hours of load, which is comparable to 'available_time'.

Properties: Export-Only Field

load_policy -- an extension selector of model Resource_Plan

This has the same value as 'resource.load_policy'. This extension defines how the Operation Loads placed on this Resource are planned. The associated fields describe the rules and restrictions on the Loads and the Problems that could occur due to usage of this Resource's capacity.

Properties: extension_same_as_model=Resource Export-Only Field

Extensions: SIMPLE_CONSOLIDATION, INFINITE_USE, EXCLUSIVE_USE, SHARED_USE.

balance_time (Date_Range, Logical, Logical) -- a Void field of model Resource_Plan

Apply the intelligence of the 'load_policy' to move the 'load_plans' so as to balance the 'load_time' with the 'available_time' in the specified Date_Range. The three Logical arguments specify whether or not to try moving earlier, moving later, and/or moving off to an alternate, respectively.

Properties: command=True Export-Only Field

balance_sid_time (Date_Range, Logical, Logical) -- a Void field of model Resource_Plan

Apply the intelligence of the 'load_policy' to move the 'load_plans' so as to balance the 'load_sid_time' with the 'capacity' in the specified Date_Range. The three Logical arguments specify whether or not to try moving earlier, moving later, and/or moving off to an alternate, respectively.

Properties: command=True Export-Only Field

move (Load_Plan, Logical, Logical) -- a Void field of model Resource_Plan

Apply the intelligence of the 'load_policy' to move the 'load_plans' so as to balance the 'load_sid_time' with the 'capacity' in the specified Date_Range. The three Logical arguments specify whether or not to try moving earlier, moving later, and/or moving off to an alternate, respectively.

Properties: command=True Export-Only Field

previous_gap (Load_Plan) -- a Date field of model Resource_Plan

The Date of the latest gap in load before the argument 'load_plan's end Date, excluding the load due to the argument 'load_plan' itself. There may not be sufficient gap to perform the entire argument 'load_plan' without Problem, but at least a portion at the end of the 'load_plan' can be performed without Problem.

Properties: Export-Only Field

next_gap (Load_Plan) -- a Date field of model Resource_Plan

The Date of the earliest gap in load after the argument 'load_plan's start Date, excluding the load due to the argument 'load_plan' itself. There may not be sufficient gap to perform the entire argument 'load_plan' without Problem, but at least a portion at the start of the 'load_plan' can be performed without Problem.

Properties: Export-Only Field

size -- an extension selector of model Resource_Plan

This has the same value as 'resource.size'. It defines the size limits on the loads that can be placed on this Resource. The associated fields describe the size (volume, weight) limits of this Resource over time. The size fields indicate how much Load can be handled at once. For example, if the size limit is "2 tons", then up to 2 tons of Loads may be simultaneously processed.

Properties: extension_same_as_model=Resource Export-Only Field

size_profile (Date_Range) -- a List(Profile_Quantity) field of model Resource_Plan

A List of all changes in the available size profile during the specified finite

Date_Range.

Properties: Export-Only Field

size_usage_profile (Date_Range) -- a List(Profile_Quantity) field of model Resource_Plan

A List of all changes in the profile of usage of size (the size of Loads) during the specified finite Date_Range.

Properties: Export-Only Field

available_sized_time (Date_Range) -- a Quantity field of model Resource_Plan
The total actual hours that this Resource is planned to be available (efficiency > 0%) during a particular Date_Range, multiplied by the size that is available during that time.

This is based on actual hours, not efficiency-adjusted "standard hours". See the sized_capacity for efficiency-adjusted standard hours.

Properties: Export-Only Field

sized_capacity (Date_Range) -- a Quantity field of model Resource_Plan
The total efficiency-adjusted "standard hours" of capacity available from this Resource during a particular Date_Range, multiplied by the size that is available during that time.

For actual hours of availability, see 'available_time'; for standard hours, see 'capacity'; and for sized hours, see 'available_sized_time'.

Properties: Export-Only Field

load_sized_time (Date_Range) -- a Quantity field of model Resource_Plan
The total actual hours of load that is planned on this Resource during a particular Date_Range, multiplied by the size of that load over that time. This excludes self-imposed setup_change and maintenance time. For comparison, 'load_sized_time(pd)' is comparable to 'available_sized_time(pd)'. See 'load_sized_std_time' for a similar sized measure of load based on "standard hours", which is comparable to 'sized_capacity'.

Properties: Export-Only Field

load_sized_std_time (Date_Range) -- a Quantity field of model Resource_Plan
The total "standard hours" of load that is planned on this Resource during a particular Date_Range, multiplied by the size of that load over that time. This excludes self-imposed setup_change and maintenance time. For comparison, 'load_sized_std_time(pd)' is comparable to 'sized_capacity(pd)'. See 'load_sized_time' for a similar sized measure of load based on total actual hours of load, which is comparable to 'available_sized_time'.

Properties: Export-Only Field

balance_sized_time (Date_Range, Logical, Logical, Logical) -- a Void field of model Resource_Plan

Apply the intelligence of the 'load_policy' to move the 'load_plans' so as to balance the 'load_sized_time' with the 'available_sized_time' in the specified Date_Range. The three Logical arguments specify whether or not to try moving earlier, moving later, and/or moving off to an alternate, respectively.

Properties: command=True Export-Only Field

balance_sized_std_time (Date_Range, Logical, Logical, Logical) -- a Void field of model Resource_Plan

Apply the intelligence of the 'load_policy' to move the 'load_plans' so as to balance the 'load_sized_std_time' with the 'sized_capacity' in the specified Date_Range. The three Logical arguments specify whether or not to try moving earlier, moving later, and/or moving off to an alternate, respectively.

Properties: command=True Export-Only Field

maintenance -- an extension selector of model Resource_Plan

Defines how maintenance is specified for this Resource. The associated fields describe when maintenance should be performed and what Operation is used to perform it. It may describe both major and minor maintenance Operations. It may base timing either total time, loaded time, setup time, regular calendar intervals, or many other criteria.

Properties: extension_same_as_model=Resource Export-Only Field

problems, problems (Date_Range), problems (Problem_Category), problems (Date_Range, Problem_Category) -- a List(Problem) field of model Resource_Plan

The Problems detected with this Resource_Plan. If passed a Date_Range, only the Problems whose dates 'overlap' are returned. If passed a Problem_Category, only the Problems with that 'category' are returned.

Properties: Export-Only Field

Model	Resource_Plan Model
-------	---------------------

problem_categories - - a List[Problem_Category] field of model Resource_Plan
 For each different category of Problem in problems, a Problem_Category is added to this list. It gives you the name of the category (in various forms) plus a list of just the Problems of that category. These sublists are often much easier to deal with than the full problems list.
 Properties: Export-Only Field

problem_time (Date_Range) - - a Time field of model Resource_Plan
 The total actual hours of Problems with the load that is planned on this Resource during a particular Date_Range. For comparison, problem_time(pd) is comparable to available_time(pd). See problem_std_time for total "standard hours" of Problems, which is comparable to capacity.
 Properties: Export-Only Field

problem_std_time (Date_Range) - - a Time field of model Resource_Plan
 The total "standard hours" of Problems with the load that is planned on this Resource during a particular Date_Range. For comparison, load_std_time(pd) is comparable to capacity(pd). See load_time for total actual hours of load, which is comparable to available_time.
 Properties: Export-Only Field

problem_sized_time (Date_Range) - - a Quantity field of model Resource_Plan
 The total actual hours of Problems with the load that is planned on this Resource during a particular Date_Range, multiplied by the size of those Problems over that time. For comparison, problem_sized_time(pd) is comparable to available_sized_time(pd). See problem_sized_std_time for a similar sized measure of load based on "standard hours", which is comparable to sized_capacity.
 Properties: Export-Only Field

problem_sized_std_time (Date_Range) - - a Quantity field of model Resource_Plan
 The total "standard hours" of Problems with the load that is planned on this Resource during a particular Date_Range, multiplied by the size of those Problems over that time. For comparison, problem_sized_std_time(pd) is comparable to sized_capacity(pd). See problem_sized_time for a similar sized measure of load based on total actual hours of load, which is comparable to available_sized_time.
 Properties: Export-Only Field

Model	Resource_Plan Model
-------	---------------------

resolve (Problem, Logical, Logical, Logical) - - a Void field of model Resource_Plan
 Resolve the Problems in the argument List, or in the specified Date_Range. The three Logical arguments specify whether or not to try moving earlier, moving later, and/or moving off to an alternate, respectively.
 Properties: command=True Export-Only Field

owner - - a Site_Plan field of model Resource_Plan
 The Site_Plan for which this Resource_Plan is a component.
 Properties: Export-Only Field

5.1.3.1.4.1 load_policy submodels of model Resource_Plan
5.1.3.1.4.2 Consolidation Model

Consolidation -- a submodel of model Resource_Plan

Consolidation is sub-model of resource_plan created to group load_plans together. As of today, they are supported for SIMPLE_CONSOLIDATION load_policy only.

The Consolidation model has fields that references these models :
Resource_Plan.

These models have a field that is a Consolidation model :
SIMPLE_CONSOLIDATION, UNCOORDINATED,
CONSOLIDATION_OVERSIZE, CONSOLIDATION_UNDERSIZE.

The key field for this model is name
This model may be extended with user-defined fields.

name -- a Symbol field of model Consolidation
The name of this Consolidation.
Default: None -- this is a key field

operation_plans -- a List(Operation_Plan) field of model Consolidation
The list of Operation_Plan in this Consolidation.
Properties: Export-Only Field

inc (Operation_Plan) -- a Void field of model Consolidation
Add an opplan to this consolidation Adding an opplan can create UNCOORDINATED or
CONSOLIDATION_OVERSIZE problem.
Properties: command=True Export-Only Field

dec (Operation_Plan) -- a Void field of model Consolidation
remove an opplan from the consolidation Removing an opplan can cause UNCONSOL-
IDATED and/or CONSOLIDATION_UNDERSIZE problem.
Properties: command=True Export-Only Field

owner -- a Resource_Plan field of model Consolidation
The owner of this Consolidation
Properties: Export-Only Field
5.1.3.1.5 Operation_Plan Model

Operation_Plan -- a submodel of model Site_Plan

The plan for performing an Operation. It includes start and end Dates for the activi-
ties, the Resources that will be loaded and when, the Buffers that will be consumed
from or produced into and when, and all other information specific to a particular
Operation.

If the Operation_Plan has been released to be performed, then it will be given a
'release_name'. At that point, changes to the plan are only made due to feasibility
Problems related to state information. State information is recorded in the
Operation_Plan, but may come indirectly from Operation_State models.

The state fields are designed to support the "minimal transaction" philosophies, such
as Just-In-Time (JIT). As time marches forward, we simply assume everything is
going according to the Plan, unless told otherwise in the form of newstate settings.

For similar reasons, plus the fact that state information often comes from other tools,
we do not demand the user specify the Operation_Plan or even a 'release_name' in
order to report state information. See the Operation_State model for that functionality.

The model has selectors:
 motive, process.

The Operation_Plan model has these submodels :
Load_Plan, Flow_Plan.

The Operation_Plan model has fields that references these models :
Operation, Load_Plan, Flow_Plan, Operation_Plan, Site_Plan.

These models have a field that is a Operation_Plan model :
Operation_Plan, Buffer_Plan, LINK, Operation_State, Item_Request,
Load_Plan, Flow_Plan, Item_Acceptance, Item_Promise,
REQUEST_NOT_PLANNED, REQUEST_PLANNED_LATE,
REQUEST_PLANNED_EARLY, REQUEST_PLANNED_SHORT,
REQUEST_PLANNED_EXCESS, PROMISE_NOT_PLANNED,
PROMISE_PLANNED_LATE, PROMISE_PLANNED_EARLY,
PROMISE_PLANNED_SHORT, PROMISE_PLANNED_EXCESS,
ACCEPTANCE_NOT_PLANNED, ACCEPTANCE_PLANNED_LATE,
ACCEPTANCE_PLANNED_EARLY, ACCEPTANCE_PLANNED_SHORT,
OVER_RESTRICTION, EXPEDITED,
PLANNED_BEFORE_CURRENT, UNRELEASED, NEEDS_RELEASE,

Models	Operation_Plan Model
--------	----------------------

INCONSISTENT_OPPLAN, SUPPLY_PLANNED_LATE, SUPPLY_PLANNED_EARLY, SUPPLY_PLANNED_SHORT, SUPPLY_PLANNED_EXCESS, SUPPLY_PROMISED_LATE, SUPPLY_PROMISED_EARLY, SUPPLY_PROMISED_SHORT, SUPPLY_PROMISED_EXCESS.

The key field for this model is operation
This model may be extended with user-defined fields.

operation -- *a Operation field of model Operation_Plan*

The Operation being planned by this Operation_Plan. The Operation defines what is to be done by this Operation_Plan.

Default: None -- this is a key field

remark -- *a String field of model Operation_Plan*

Comments about this specific plan to perform this operation.

Default: none

rank -- *a Number field of model Operation_Plan*

The 'rank' of this Operation_Plan, which is a relative measure of importance, priority, or significance. Its value can be dynamic (changed by the planning algorithms); its effects can be dynamic (different algorithms and different Strategies may treat ranks differently); and overall its semantic is user-controllable. This is used to lock Operation_Plan during strategy runs when used in conjunction with Strategy_Lock model.

Rank can be set on a top Operation_Plan at the time of its creation, using the Plan's *set_default_operation_plan_rank* expression.

Default: 0

released -- *a Logical field of model Operation_Plan*

If "true", then this Operation_Plan has been released to be performed. Once released, it is subject to greater constraints. Its motive and quantity will not be changed, and it will be executed as soon as feasible. The material assigned to it is considered "allocated" and will not be "re-allocated". Of course, users may change anything about a released Operation_Plan they desire. When the Operation_Plan is for an operation which is part of an operation hierarchy, (it has sub-operations, or it is itself a sub-operation, or both), the entire hierarchy is released together.

Default: false

Models	Operation_Plan Model
--------	----------------------

release_name -- *a Symbol field of model Operation_Plan*

The *release_name* used to identify an Operation_Plan when it is being processed. This is often called an "order id". This need not be used -- for example, repetitive manufacturers often do not need to bother with a *release_name*. But even in that case, a *release_name* will be generated when released is set 'true' (see the description of the *release_name_expression* field). If the Operation_Plan is was not released when the *release_name* is set, the Operation_Plan will be released. As with released Operation_Plan's, if the Operation_Plan is part of a hierarchy, the *release_name* refers to the entire hierarchy.

Default: none

release_fence_date -- *a Date field of model Operation_Plan*

Returns the date this op_plans *release_fence* becomes effective.

Default: 0

Properties: Export-Only Field

release_soon_fence_date -- *a Date field of model Operation_Plan*

Returns the date this op_plans *release_soon_fence* becomes effective.

Default: 0

Properties: Export-Only Field

use_alternate (Operation_Plan), *use_alternate* (Operation_Plan, Operation) --

a Void field of model Operation_Plan

Replan the Operation_Plan to try to use a different sub_operation_plan.

Properties: command=True Export-Only Field

motive -- *an extension selector of model Operation_Plan*

The purpose this Operation_Plan was created. It is one of: PRODUCE (produce 'quantity' units into a Buffer), CONSUME (consume 'quantity' units from a Buffer), DELIVER (deliver 'quantity' units to an address), RECEIVE (receive 'quantity' units from an address).

This field is read-only.

Properties: Export-Only Field

Extensions:

PRODUCE, CONSUME, DELIVER, RECEIVE,

units -- *a Number field of model Operation_Plan*

The number of units of this Operation planned. Note that, due to yields, this Quantity may vary as the Operation_Plan moves, even when the motive remains the same.

Models**Operation_Plan Model**

This is always unitless, of course -- the Number of units of this Operation. See 'quantity' for the same value returned in the Operation's 'preferred_measure'.
Default: computed from the Operation

quantity -- *a Quantity field of model Operation_Plan*
The units of this Operation planned. Note that, due to yields, this Quantity may vary as the Operation_Plan moves, even when the motive remains the same.

This Quantity is converted to the 'unit' of this Operation.
Default: computed from the Operation

std_time -- *a Time field of model Operation_Plan*
The standard Time required by this Operation_Plan to fulfill the 'motive'. This is computed by multiplying the standard Time computed by the 'operation' with the 'expedite' Percentage.

Setting this field actually sets the 'expedite' field appropriately to give the specified result. An EXPEDITED Problem will be raised if 'expedite' is set to other than 100%.
Default: computed from the Operation

expedite -- *a Percentage field of model Operation_Plan*
Expedite values less than 100% specify a reduction in time for a particular operation plan with respect to the standard time specified by the operation. When the expedite value is greater than 100% then the operation plan takes longer. To define an operation that takes only half as long as the standard, set expedite to be 50%. Stated another way, the expedite number is 100% divided by the speed increase you desire for this particular operation plan. For 3 times faster than normal, set expedite to 33%.

The 'std_time' member will reflect the operation's standard Time multiplied by the expedite Percentage. If the 'operation' specifies the standard Time to be 8 hours and the expedite Percentage is set to "25%", then the 'std_time' will be 2 hours.

The actual time for the operation will reflect the resource loading efficiency (assuming that the process actually loads a resource -- some don't).
Default: 100%

dates -- *a Date_Range field of model Operation_Plan*
The Date_Range during which this Operation_Plan is planned to be performed. So, 'dates.start' is the start Date, 'dates.end' is the end Date. And 'dates.time' is the actual duration of this Operation_Plan. It is primarily affected by the efficiency of each of the Resources this Operation_Plan is loading. The minimum efficiency of the Resources at any given Date is the efficiency of this Operation_Plan at that Date.

Models**Operation_Plan Model**

Properties: Export-Only Field

hint -- *a Restriction field of model Operation_Plan*
A temporary Restriction on this Operation_Plan. Setting this causes the Operation_Plan to be moved accordingly (if possible). For example, setting it to "start after 97-05-01 12:00" will move the planned dates to "97-05-01 12:00". If it is already after "97-05-01 12:00", then it will not move. Note that the engine will accept second granularity, but requires at least minute granularity.

The default value is "[unspecified]" (however, the engine will accept [] and change it to [unspecified]).

The hint can have one of the following forms: "start before <date>", "start after <date>", "end before <date>", "end after <date>", "first in <periods>", "last in <periods>", or "[unspecified]".

The "start before <date>" and "start after <date>" actually executes as "start at" and "end before". The "end before" and "end after" both execute as "end at".

The "first in <periods>" option is actually a choice between "end first in <date_range>" and "start first in <date_range>". If the user types "first in <date_range>", it automatically changes to "start first in <date_range>". The "start first in <date_range>" sets the start of the operation plan as early in the date_range as possible. The "end first in <date_range>" sets the end of the operation plan as early in the date_range as possible.

The "last in <periods>" option is actually a choice between "end last in <date_range>" and "start last in <date_range>". If the user types "last in <date_range>", it automatically changes to "end last in <date_range>". The "start last in <date_range>" sets the start of the operation plan as late in the date_range as possible. The "end last in <date_range>" sets the end of the operation plan as late in the date_range as possible.

Note that this setting is temporary, but once the engine attempts to follow the hint, the engine never again attempts to follow the hint until the user sets (types) it again.
Default: empty (within the infinite Date_Range)

locked_as_planned -- *a Logical field of model Operation_Plan*
A flag which indicates if this Operation_Plan cannot be edited (moved, resized, etc.). Attempts to replace a locked_as_planned operation plan will fail. If this results in a situation where a locked_as_planned operation plan causes the plan to be inconsistent due to an edit of a model, an INCONSISTENT_OPLAN problem is raised. See the description of the INCONSISTENT_OPLAN problem for more detailed information.

Default: FALSE

load_plans - - *a list of Load_Plan submodels of model Operation_Plan*

These load_plans specify the Resources to be loaded for the duration of this Operation.

flow_plans - - *a list of Flow_Plan submodels of model Operation_Plan*
The Item Flow_Plan generated for the duration of this Operation.

all_consuming_flow_plans - - *a List(Flow_Plan) field of model Operation_Plan*
Return all consuming Flow_Plan in this Operation_Plan and its sub_operation_plans.

Properties: Export-Only Field

all_producing_flow_plans - - *a List(Flow_Plan) field of model Operation_Plan*
Return all producing Flow_Plan in this Operation_Plan and its sub_operation_plans.
Properties: Export-Only Field

sub_operation_plans - - *a List(Operation_Plan) field of model Operation_Plan*
The Operation_Plan that model smaller portions/phases of this Operation_Plan.
Properties: Export-Only Field

super_operation_plan - - *a Operation_Plan field of model Operation_Plan*
The routing, alternates, or other containing Operation_Plan of which this Operation_Plan is a part. This may be a coordinating Operation_Plan that does not really correspond to the Operations in the Site.

This may be "nonexistent" if this Operation was directly planned by either buffer_plan or item_promise.

Properties: Export-Only Field

top_operation_plan - - *a Operation_Plan field of model Operation_Plan*
The topmost Operation_Plan containing this Operation_Plan. It is equivalent to calling super_operation_plan repeatedly until it returns nonexistent.
Properties: Export-Only Field

process - - *an extension selector of model Operation_Plan*

This is the same as 'operation.process'. It defines the fields that are specific to this Operation's process.

Properties: extension_same_as_model=Operation Export-Only Field

Extensions:

ALTERNATES, PRIMARY, ALTERNATES, PROPORTIONAL, EFFECTIVE, CALENDAR, DELAY_ONLY_FIXED, DELAY_ONLY_BASIC, BASIC, CALENDARS, FIXED_TIME, TIME_MULTIPLE, BASIC, BASIC, DELAYED, REQUEST_FIXED, REQUEST_FIXED_WITH_ANALYSIS, ROUTING,

problems, problems (Date_Range) , problems (Problem_Category) , problems (Date_Range, Problem_Category) - - *a List(Problem) field of model Operation_Plan*

The Problems detected with this Operation_Plan. If passed a Date_Range, only the Problems whose dates 'overlap' are returned. If passed a Problem_Category, only the Problems with that 'category' are returned.

Properties: Export-Only Field

problem_categories - - *a List(Problem_Category) field of model Operation_Plan*
For each different 'category' of Problem in 'problems', a Problem_Category is added to this list. It gives you the name of the 'category' (in various forms) plus a list of just the Problems of that 'category'. These sublists are often much easier to deal with than the full 'problems' list.

Properties: Export-Only Field

split (Item, Quantity, Logical) , split (Quantity) , split (Quantity, Restriction) - - *a Operation_Plan field of model Operation_Plan*
Split will allow users to split this operation plan in to two considering given parameters. It will return the newly created second operation plan as a result of split.
Properties: command=True Export-Only Field

owner - - *a Site_Plan field of model Operation_Plan*

Properties: Export-Only Field

Models	Load_Plan Model
--------	-----------------

5.1.3.1.5.1 Load_Plan Model

Load_Plan -- *a submodel of model Operation_Plan*

The Resource_Plan planned to be loaded by the 'owner' Operation_Plan, for placement of a particular load on a Resource, typically by an Operation or Buffer. A Load has start and end dates (and thus duration) and a Quantity. It knows the Resource it is loading, as well as the Operation or Buffer placing the load.

The Load_Plan model has fields that references these models :
Load, Resource_Plan, Operation_Plan.

These models have a field that is a Load_Plan model :
Operation_Plan.

The key field for this model is load

load -- *a Logical field of model Load_Plan*

The Load of the 'owner' operation for which this is a plan. The Load specifies a Skill which could include many Resources. This Load_Plan specifies which of those Resources will be loaded.

Default: None -- this is a key field
Properties: Export-Only Field

resource_plan -- *a Resource_Plan field of model Load_Plan*

The Resource that is being loaded. When this is set by the user, the hint is set to "ON" to indicate the user is temporarily restricting that choice, as a hint to the automated algorithms. To make this setting stick, set lock to "true".
Default: [unspecified]

dates -- *a Date_Range field of model Load_Plan*

The dates during which this Load is planned to occur. These 'dates' will be within the 'owner' Operation_Plan's dates.
Properties: Export-Only Field

hint -- *a Restriction field of model Load_Plan*

A temporary Restriction on this Operation_Plan. Setting this causes the Operation_Plan to be moved accordingly (if possible). For example, setting it to "start after 05-01" will move the planned 'dates' such that this Load_Plan occurs on or after "05-01".

Models	Load_Plan Model
--------	-----------------

See the hint field in the Operation_Plan for more information.

Note that setting this hint will generally set the 'owner:hint', though to possibly a somewhat adjusted value designed to reflect the difference between restricting the timing of the Load_Plan rather than the whole Operation_Plan.

Note that relaxing the Restriction will not cause the Operation_Plan to move -- it only moves if it does not already satisfy the Restriction. Thus, setting the hint back to unrestricted will not cause the Operation_Plan to move back to where it was.
Default: empty (within the infinite Date_Range)

hint_on -- *a Logical field of model Load_Plan*

This is a temporary specification that the 'owner' Operation_Plan should continue to load this Resource_Plan. If "true", then effectively 'locked' is set temporarily, such that it will not be moved to an alternate. Note, though, unlike 'locked', this value can be ignored and/or reset.
Default: false

lock_on -- *a Logical field of model Load_Plan*

If "true", then the 'owner' Operation_Plan is locked onto this 'resource_plan'. It cannot be moved to an alternate.
Default: false

use_alternate, use_alternate (Resource_Plan) -- *a Void field of model Load_Plan*

Replan the 'owner' Operation_Plan to use a different Resource_Plan, if possible. If a Resource_Plan is specified, then switch to that Resource_Plan, if possible, and set hint_on to "true". When given a Resource_Plan argument, it is the same as setting 'resource_plan' field.
Properties: command=True Export-Only Field

owner -- *a Operation_Plan field of model Load_Plan*

The Operation_Plan placing this Load_Plan on 'resource'.
Properties: Export-Only Field

5.1.3.1.5.2 Flow_Plan Model

Flow_Plan -- a submodel of model Operation_Plan

The plan for flow of items between Buffers and Operations. A Flow_Plan has a Quantity, a Buffer, an Operation, and a direction ('produced' or not).

The Flow_Plan model has these submodels :

Lot_Flow.

The Flow_Plan model has fields that references these models :

Flow, Buffer_Plan, Lot_Flow, Operation_Plan.

These models have a field that is a Flow_Plan model :

Operation_Plan, Lot_Flow.

The key field for this model is Flow

Flow -- a Flow field of model Flow_Plan

The Flow of the owner's operation for which this is a plan. The Flow specifies how much flows between the Buffer and the Operation. This Flow_Plan specifies how much will flow for this particular Operation_Plan.

Default: None -- this is a key field

Properties: Export-Only Field

buffer_plan -- a Buffer_Plan field of model Flow_Plan

The Buffer_Plan that is being produced into or consumed from by the owner's Operation_Plan.

Properties: Export-Only Field

produced -- a Logical field of model Flow_Plan

If "true", then the owner's Operation is producing 'quantity' items into the buffer. If "false", then the owner's Operation is consuming 'quantity' items from the buffer. This is specified by the flow's 'usage_policy' and related fields.

Properties: Export-Only Field

dates -- a Date_Range field of model Flow_Plan

The dates during which this Flow is planned to occur. These 'dates' will be within the owner's Operation_Plan's dates.

Properties: Export-Only Field

quantity, quantity (Date) , quantity (Date_Range) -- a Quantity field of model Flow_Plan

The Quantity that is being consumed by the owner's Operation_Plan. Note that if the field 'produced' is yes, then this Quantity will be negative -- this field is always the amount consumed from the Buffer.

If no argument is passed, then this returns the quantity consumed during the entire duration of the operation. If a single Date is passed, then it will give the Quantity consumed by that Date. If a pair of Dates is passed, then it will give the Quantity consumed between those two Dates. (Passing one Date is equivalent to passing the operation's start_date and that Date; passing nothing is equivalent to passing the operation's start and end dates.)

The Quantity is converted to the unit of the buffer_plan.buffer.

Default: 0

lots -- a list of Lot_Flow submodels of model Flow_Plan

The lots being consumed from or produced into a Buffer.

upstream_flow_plans -- a List(Flow_Plan) field of model Flow_Plan

A list of all Flow_Plans that feed this one, either through a Buffer_Plan or an Operation_Plan. If this is a 'produced' Flow_Plan, then this is a list of the Flow_Plans consumed by the Operation_Plan that will produce this Flow_Plan. If this is not a 'produced' Flow_Plan, then this is a list of the Flow_Plans that will produce the items that this Flow_Plan will consume from the Buffer_Plan.

Properties: command=True Export-Only Field

downstream_flow_plans -- a List(Flow_Plan) field of model Flow_Plan

A list of all Flow_Plans that are fed by this one, either through a Buffer_Plan or an Operation_Plan. If this is a 'produced' Flow_Plan, then this is a list of the Flow_Plans that will consume the items from the Buffer_Plan that this Flow_Plan produces. If this is not a 'produced' Flow_Plan, then this is a list of the Flow_Plans that are produced by the Operation_Plan that consumes this Flow_Plan.

Properties: command=True Export-Only Field

owner -- a Operation_Plan field of model Flow_Plan

The Operation_Plan placing this Flow_Plan on 'buffer'.

Properties: Export-Only Field

5.1.3.15.2.1 Lot_Flow Model

Lot_Flow -- a submodel of model Flow_Plan

The lots being consumed from or produced into a Buffer.

The Lot_Flow model has fields that references these models :
Lot, Flow_Plan.

These models have a field that is a Lot_Flow model :
Flow_Plan, Lot.

The key field for this model is lot

lot -- a Lot field of model Lot_Flow

The Lot of the buffer_plan that is being consumed from or produced into.

Default: None -- this is a key field

quantity -- a Quantity field of model Lot_Flow

The Quantity of the 'lot' that is being consumed; this may be less than the total Quantity of the 'lot', depending upon the Flow's usage_policy.

The Quantity is negative if this Flow_Plan is producing into the 'lot'. In that case, this Quantity will be the total Quantity of the 'lot' (but negative).

The Quantity is converted to the unit of the buffer_plan buffer.

Default: 0

consuming_flow_plan -- a Flow_Plan field of model Lot_Flow

The Flow_Plan that consumes the Lot making up this Lot_Flow.

Properties: Export-Only Field

upstream_lot_flows -- a List(Lot_Flow) field of model Lot_Flow

A list of all Lot_Flows that feed this one skipping over operations. Recursing with this function can be used for keeping track of Lots and costs for a particular item.

Properties: command=True Export-Only Field

downstream_lot_flows -- a List(Lot_Flow) field of model Lot_Flow

A list of all Lot_Flows that are fed by this one skipping over operations. Recursing with this function can be used for keeping track of the items that Lot ends up in, and the costs associated with that.

Properties: command=True Export-Only Field

owner -- a Flow_Plan field of model Lot_Flow

Properties: Export-Only Field

5.1.3.1.6 Operation_State Model

Operation_State -- a submodel of model Site_Plan

Operation_State is an indirect mechanism for reporting the state of Operation_Plans. It is not normally used interactively -- it is much easier to access the Operation_Plan directly. The Operation_State is primarily intended to map state information provided by other systems (ERP, MRP II, SFC, MES, etc.) to the appropriate Operation_Plan.

The 'state_spec' extension specifies how the state information itself is expressed. The 'identifier' extension specifies how the Operation_Plan that this State pertains to is identified. The 'operation_plans' List is all Operation_Plans that match the criteria. The 'operation_plan' is the Operation_Plan that this Operation_State is applied to. It can be set by the user, and need not match any that appear in 'operation_plans'.

If zero or more than one Operation_Plan matches, then 'operation_plan' will be initially set to [unspecified] and an UNIDENTIFIED_OP_STATE Problem will be raised. The 'operation_plan' must be set interactively to eliminate the Problem.

Operation_State is usually only interacted with in order to correct errors or to aid or correct the identification of the Operation_Plan. Important: When an op state is imported into SCP, it will not be attached to an op plan immediately. Currently Operation_States are attached to Operation_Plans as a side effect of problem list examination, either directly via a call to Plan.problems, or indirectly via strategy execution. To avoid unexpected behavior, use Plan.problems.count as soon as wip is imported or when the operation_plan field is set.

The model has selectors:

identifier, state_spec.

The Operation_State model has fields that references these models :
Operation_Plan, Site_Plan.

These models have a field that is a Operation_State model :
LINK, UNIDENTIFIED_OP_STATE.

The key field for this model is identifier

Models	Operation_State Model
--------	-----------------------

This model may be extended with user-defined fields.

operation_plan - - *a Operation_Plan field of model Operation_State*
 The Operation_Plan that this is reporting about. Once set (or computed) to other than the default (unspecified), the Operation_State will appear in that Operation_Plan's operation_states List and will be used to compute its state.
 Default: [unspecified]

unattach - - *a Void field of model Operation_State*
 Causes the Operation_State to unattach itself from the currently selected operation_plan. If the currently selected operation_plan is unspecified then nothing will happen.
 Properties: command=True Export-Only Field

operation_plans - - *a List(Operation_Plan) field of model Operation_State*
 The Operation_Plans that are identified by the fields associated with the Identifier extension.
 Properties: Export-Only Field

Identifier - - *an extension selector of model Operation_State*
 Specifies how to identify the Operation_Plan that this Operation_State pertains to. For example, it may specify the release_name of a released Operation_Plan, or it may specify the release_name of a released super-Operation and a sub-Operation name, or it may specify a super-Operation name and a sub-Operation name and expect the earliest that fits to be identified, or it may specify an Item name and Lot name, or many other possibilities. Currently, however, the only available Identifier is EARLIEST.
 Default: None - - this is a key field

Extensions:
 EARLIEST.

date - - *a Date field of model Operation_State*
 The Date at which this Operation_State was accurate.
 Default: today

state_spec - - *an extension selector of model Operation_State*
 Specifies the fields and how they will be used to specify the state of the operation_plan. For example, a shop floor control system may give cumulative completed quantities, the amount in-process, the percent-complete, the hours completed, the hours remaining, etc. This extension supports easy mapping of such varied input forms into the Operation_Plan model.
 Default: COMPLETED

Models	Request Model
--------	---------------

Extensions:
 STARTED, COMPLETED, IN_FRONT,

owner - - *a Site_Plan field of model Operation_State*

Properties: Export-Only Field
 5.1.3.1.7 Request Model

Request - - *a submodel of model Site_Plan*

The Request, Delivery_Request, and Item_Request models together model requests from one Site to another. The parallel Promise, Delivery_Promise, and Item_Promise models together model the supplying Site's commitment back to the requesting Site. These models together implement what is traditionally called "Order Management" or "Demand Management". Simplified variations of those models are often called "orders".

The term "order" is intentionally not used in the model and field names. Order is used to mean very different things in different industries, in different companies in the same industry, and even in different groups in the same company. There are demand orders, sales orders, purchase orders, blanket orders, work orders, maintenance orders, shop orders, and manufacturing orders, many of which are very different from each other. And each of those may mean subtly or drastically different things for different people. To avoid confusing terminology then, we have chosen to avoid the term "order" in model and field names, though we will use it in explanations such as this one.

An Item_Request can model a typical sales-order line-item. The Delivery_Request can model an order where all the line items are to be delivered together. A Request can model an order containing groups of line items that are to be delivered at different Dates, or can model a long-term contract or agreement for orders to be placed in the future.

Fundamentally, a Request is a set of Delivery_Requests that are issued together and a Promise is to be offered for the whole. If the Delivery_Promises or Item_Promises are to be offered separately, then they should be placed in separate Requests and Promises.

For long-term negotiated contracts, the 'delivery_policy' can define how Delivery_Requests are generated and adjusted. This provides similar functionality to Forecast's generation of forecast Requests and adjustment of those Requests as actual Requests arrive. However, in this case it is to reflect Delivery_Requests that are part of a larger agreement between two Sites. This is often referred to as a "blanket order" or a "contract".

The supplier's half of the order (e.g., the agreed-to due date) is modeled in the parallel Promise models (Promise, Delivery, Promise, and Item_Promise). These are separated from the corresponding Request models for protocol reasons -- only the customer should be modifying the Request, and only the supplier should be modifying the Promise.

In addition to the model separation, there are numerous fields designed to support the typical order protocols that are needed in real situations. The basic flow is as follows: (1) the customer issues a Request, (2) the supplier offers a Promise to that Request which may differ from the Request (may even be zero), and (3a) the customer accepts the offer (adjusting the Request to match the Promise), or (3b) the customer accepts the offer but also queues the Request for more (assuming the Promise was less than Requested), or (3c) the customer deletes the Request. Deadlines may be imposed on these steps. When a Request is issued it may specify a deadline for receiving a Promise. And the Promise may specify a deadline for acceptance (the offer is only good until that deadline).

This protocol is maintained by simple Date fields that keep track of "when things happened. By simply comparing the Dates, the state of the protocol is known. The fields are: 'date_issued' (set by step 1), 'promise_date_offered' (set by step 2), 'date_queued' (set by 3b), and 'date_accepted' (set by 3a and 3b). The deadlines are 'promise_by' and 'promise_accept_by' (note that the Request's 'accept_by' is just the desired deadline).

If 'date_issued' is after 'promise_date_offered', then the Request has not been responded to, and a PROMISE_NOT_OFFERED Problem will exist (its severity depends upon 'request_promise_by'). This is resolved by step (2).

If 'promise_date_offered' is at or after 'date_issued' and 'date_accepted' is before 'promise_date_offered', then the Request has been offered a Promise, but the Promise has not been accepted. A PROMISE_NOT_ACCEPTED Problem will exist (its severity depends upon 'promise_accept_by'). This is resolved by step (3b).

If 'date_accepted' is at or after 'promise_date_offered' which is at or after 'date_issued', then the Promise has been offered and accepted and is now binding on both parties.

If 'date_queued' is at or after 'promise_date_offered' which is at or after 'date_issued', then the Request has been queued for later consideration and a REQUEST_QUEUED Problem will exist.

The model has selectors:
delivery_policy, delivery_naming.

The Request model has these submodels:
Delivery_Request.

The Request model has fields that reference these models:
Delivery_Request, Promise, Seller_Plan, Forecast, Site_Plan.

These models have a field that is a Request model:

Forecast, LINK, Promise, SUPPLIER, Delivery_Request,
REQUEST_NOT_PLANNED, REQUEST_PLANNED_LATE,
REQUEST_PLANNED_EARLY, REQUEST_PLANNED_SHORT,
REQUEST_PLANNED_EXCESS, PROMISE_NOT_PLANNED,
PROMISE_PLANNED_LATE, PROMISE_PLANNED_EARLY,
PROMISE_PLANNED_SHORT, PROMISE_PLANNED_EXCESS,
ACCEPTANCE_NOT_PLANNED, ACCEPTANCE_PLANNED_LATE,
ACCEPTANCE_PLANNED_EARLY, ACCEPTANCE_PLANNED_SHORT,
REQUEST_PROMISED_EARLY, REQUEST_PROMISED_LATE,
REQUEST_PROMISED_EXCESS, ITEM_PROMISE_OVERPRICED,
DELIVERY_PROMISE_OVERPRICED, PROMISE_NOT_OFFERED,
PROMISE_NOT_CONFIRMED, PROMISE_NOT_ACCEPTED,
ACCEPTANCE_INCONSISTENT, REQUEST_QUEUED,
DELIVERY_REQUEST_NOT_COORDINATED,
DELIVERY_PROMISE_NOT_COORDINATED,
DELIVERY_ACCEPTANCE_NOT_COORDINATED,
SUPPLY_PLANNED_LATE, SUPPLY_PLANNED_EARLY,
SUPPLY_PLANNED_SHORT, SUPPLY_PLANNED_EXCESS,
SUPPLY_PROMISED_LATE, SUPPLY_PROMISED_EARLY,
SUPPLY_PROMISED_SHORT, SUPPLY_PROMISED_EXCESS.

The key field for this model is name

This model may be extended with user-defined fields.

name - - a Symbol field of model Request

The name of this Request. The name must be unique among Requests of the 'owner' Site_Plan.

For single-delivery Requests, this may be the "order ID". The Delivery_Request name may be the same or may be something simple like "0". For blanket orders or contracts, this is the contract ID.

Often, this name is formed from the customer name and their PO# concatenated together.

Default: None -- this is a key field

description -- *a String field of model Request*

A description of or comments about this Request.

Default: none

delivery_requests -- *a list of Delivery_Request submodels of model Request*

The individual orders that are being requested. Each Delivery_Request consists of one or more Item_Requests that are to be delivered together.

delivery_policy -- *an extension selector of model Request*

Defines what 'delivery_requests' will be created, and what should be created. Defines the forecast orders and how they change as actual orders are introduced.

The default, FIXED, specifies that there is no order variation -- the 'delivery_requests' explicitly list what is being requested. If all the 'delivery_requests' are 'promised', then (and only then) is this Request 'promised'.

Default: FIXED

Extensions:

FIXED.

delivery_naming -- *an extension selector of model Request*

Defines how Delivery_Requests generated by the delivery_policy should be named.

Default: NUMBERED

Extensions:

NUMBERED.

accept_by -- *a Date field of model Request*

The Date by which the customer_plan intends to accept or reject any Promise that is made. This is the Date on which both parties would become committed. And it is the start of the delivery_lead_time (which can affect pricing).

Note that this is just the requested 'accept_by' Date -- the actual deadline for accepting a Promise is in the Promise 'accept_by'. The supplier considers this requested Date when making the Promise, but may choose a different Date based upon the delivery lead times of its Products. Such differences are part of the negotiation of Request and Promise.

Default: promise_by + "1 day"

promise -- *a Promise field of model Request*

The Promise made in response to this Request. Note that there is always a Promise model for any Request model, but it may not be 'promised' yet. In fact, it could be rejected (Promise of zero). The Promise models either of those answers (or lack of an answer) to this Request.

Properties: Export-Only Field

promise_by -- *a Date field of model Request*

The Date by which an answer to this Request is required. This Request will automatically expire on this Date if no answer is given. The default, oo_future, indicates that the Request will not expire automatically.

Default: oo_future

date_issued -- *a Date field of model Request*

The Date that this Request is considered to have been made. If this Date is after the Dates in the 'promise', then this Request is considered to be unanswered. Thus, setting this field to 'now' essentially renews this Request.

Default: 'now'

date_accepted -- *a Date field of model Request*

The Date that the Promise offered for this Request was accepted (agreed to). Until accepted, the 'supplier' need not fill the Promise, and the customer need not accept it. Once offered, it must be accepted prior to the 'accept_by' Date in the Promise (the 'accept_by' Date in the Request is just a request).

The Promise has only been accepted when 'date_accepted' is on or after 'promise.date_offered', which itself must be on or after 'date_issued'. So, the default, oo_past, which is always prior to 'date_issued', means it has not been accepted.

Default: oo_past

date_queued -- *a Date field of model Request*

The Date that the customer_plan asked the 'owner' supplier to queue this Request. There are three possible responses to an inadequate Promise being offered. One is to withdraw (delete) the Request. One is to modify the Request and renew it by setting 'date_issued' to after 'promise.date_offered'. One is to queue the Request, by setting this field, 'date_queued', to on or after 'promise.date_offered'.

A queued Request is one that will be reconsidered if conditions at the supplier change such that new Requests can be satisfied. When that occurs, the 'owner' supplier will immediately consider any queued Requests. In that way, the customer_plan need not keep renewing a rejected Request just in case capacity has freed up.

Default: oo_past

Model	Request Model
-------	---------------

cancel - - *a Void field of model Request*
This command cancels all the delivery_requests and item_requests associated with this request. All the item_requests will be marked as cancelled. None of the item_requests or delivery_requests will be deleted. The associated delivery and receiving plans will be destroyed.
Properties: command=True Export-Only Field

plan_to_satisfy - - *a Void field of model Request*
This command creates plans to satisfy this Request. It does this by simply calling 'plan_to_satisfy' on each of its 'delivery_requests'. Thus, it is equivalent to 'delivery_requests.for_each(#.plan_to_satisfy)'.
Alternatively, 'promise.plan_to_satisfy' can be performed to call 'plan_to_satisfy' on each of the 'delivery_promises'.
Properties: command=True Export-Only Field

plan_as_requested - - *a Void field of model Request*
This command sets the plan for this Request such that it is attempting to satisfy this Request, and all of its 'delivery_requests' and 'item_requests'.
Properties: command=True Export-Only Field

seller_plan - - *a Seller_Plan field of model Request*
The Seller_Plan of the Seller that is responsible for this agreement. All 'delivery_requests' of this Request will have this or a member as its 'seller'. For example, a blanket Request could be established through an Seller organization, but it is the members that eventually make the individual Delivery_Requests.
Default: [unspecified]

forecast - - *a Forecast field of model Request*
This is the Forecast that generated this Request. If nonexistent, then this Request is an actual (customer) Request.
Properties: Export-Only Field

customer_plan - - *a Site_Plan field of model Request*
The Site_Plan of the Site placing this Request (or forecasted to be placing this Request) on the 'owner' Site_Plan. All 'delivery_requests' of this Request will have this or a member Site as its 'customer_plan'. For example, a blanket Request could be established by an ORGANIZATION Site, but it is the members that eventually make the individual Delivery_Requests.
Default: [unspecified]

Model	Request Model
-------	---------------

name_from_customer - - *a Symbol field of model Request*
The customer's name for this Request. This may be the same as 'name' or different. For example, this is often the customer's PO# (purchase order number). However, the supplier will often name Requests differently -- after all, there is no guarantee that these names (PO#s) will be unique, since they come from different customers.
Default: 'name'

delivery_name - - *a String field of model Request*
The name of the customer to which this Request should be delivered.
Default: customer.site.delivery_name

delivery_contact - - *a String field of model Request*
The name of the individual to whom this Request should be delivered.
Default: customer.site.delivery_contact

delivery_phone - - *a String field of model Request*
The phone number of the 'delivery_contact' to which this Request should be delivered.
Default: customer.site.delivery_phone

delivery_fax - - *a String field of model Request*
The fax phone number of the 'delivery_contact' to which this Request should be delivered.
Default: customer.site.delivery_fax

delivery_address - - *a String field of model Request*
The street address to which this Request should be delivered.
Default: customer.site.delivery_address

delivery_city - - *a String field of model Request*
The city (township) to which this Request should be delivered.
Default: customer.site.delivery_city

delivery_state - - *a String field of model Request*
The state (province, territory) to which this Request should be delivered.
Default: customer.site.delivery_state

delivery_country - - *a String field of model Request*
The country (nation) to which this Request should be delivered.
Default: customer.site.delivery_country

Models	Request Model
--------	---------------

delivery_postal_code -- *a String field of model Request*
The postal_code (zip code) to which this Request should be delivered.
Default: customer.site.delivery_postal_code

last_change -- *a Date field of model Request*
The Date at which the most recent change was made to this Request. This is set by setting this or any other field in this model or the 'delivery_requests'. Note that setting this directly will set it to the specified Date, which may not be the current Date.

This is used to support the 'changed_requests' functions of Site, which provide net-change Exporting of Requests.
Default: now

issued -- *a Date field of model Request*
Obsolete! This field has been renamed 'date_issued' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.

Default: now
Properties: obsolete=True

accepted -- *a Date field of model Request*
Obsolete! This field has been renamed 'date_accepted' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.

Default: oo_past
Properties: obsolete=True

queued -- *a Date field of model Request*
Obsolete! This field has been renamed 'date_queued' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.

Default: oo_past
Properties: obsolete=True

problems, **problems (Date_Range)**, **problems (Problem_Category)**, **problems (Date_Range, Problem_Category)** -- *a List(Problem) field of model Request*
The Problems associated with this Request, including any problems that are associated with the underlying Delivery_Request and Item_Request models. If passed a Date_Range, only the Problems whose 'dates' overlap are returned. If passed a Problem_Category, only the Problems with that 'category' are returned.
Properties: Export-Only Field

Models	Request Model
--------	---------------

owner -- *a Site_Plan field of model Request*
The Site_Plan upon which this Request is placed -- this Site is responsible for promising and filling the 'orders'.
Properties: Export-Only Field

5.1.3.1.7.1 Delivery_Request Model

Delivery_Request -- a submodel of model Request

Delivery_Request models a request for a set of items to be delivered together. Several Delivery_Request may make up the 'owner' Request. The Delivery_Request specifies the 'due' Date or Date_Range. The Item_Requests within the Delivery_Request specify the items and quantities.

The model has selectors:

promising_policy, fulfillment_policy;

The Delivery_Request model has these submodels :
Item_Request.

The Delivery_Request model has fields that reference these models :
Forecast_Entry, Delivery_Promise, Item_Request, Seller_Plan, Site_Plan, Problem.

These models have a field that is a Delivery_Request model :

Request, Delivery_Promise, Item_Request, REQUEST_NOT_PLANNED, REQUEST_PLANNED_LATE, REQUEST_PLANNED_EARLY, REQUEST_PLANNED_SHORT, REQUEST_PLANNED_EXCESS, PROMISE_NOT_PLANNED, PROMISE_PLANNED_LATE, PROMISE_PLANNED_EARLY, PROMISE_PLANNED_SHORT, PROMISE_PLANNED_EXCESS, ACCEPTANCE_NOT_PLANNED, ACCEPTANCE_PLANNED_LATE, ACCEPTANCE_PLANNED_EARLY, ACCEPTANCE_PLANNED_SHORT, ACCEPTANCE_PLANNED_EXCESS, REQUEST_PROMISED_LATE, REQUEST_PROMISED_EARLY, REQUEST_PROMISED_SHORT, REQUEST_PROMISED_EXCESS, ITEM_PROMISE_OVERPRICED, DELIVERY_REQUEST_OVERPRICED, PROMISE_NOT_CONFIRMED, DELIVERY_REQUEST_NOT_COORDINATED, DELIVERY_PROMISE_NOT_COORDINATED, DELIVERY_ACCEPTANCE_NOT_COORDINATED, SUPPLY_PLANNED_LATE, SUPPLY_PLANNED_EARLY, SUPPLY_PLANNED_SHORT, SUPPLY_PLANNED_EXCESS, SUPPLY_PROMISED_LATE, SUPPLY_PROMISED_EARLY, SUPPLY_PROMISED_SHORT, SUPPLY_PROMISED_EXCESS.

The key field for this model is name
This model may be extended with user-defined fields.

name -- a Symbol field of model Delivery_Request

The name of this Delivery_Request. In the common case of a single Delivery_Request in the Request, this is typically the same as the 'owner name', the order-id. Alternatively, it could be something simple like '0'. In the case of contracts, this is typically sub-order-id or delivery-id or billing-id. In the case of a Forecast Request or generated orders, this 'name' is typically generated as well.

Default: None -- this is a key field

actual -- a Logical field of model Delivery_Request

If 'true', then this is an actual (customer) Request. Otherwise, this Delivery_Request was generated either by the 'delivery_policy' of the 'owner' Request, or the 'owner' Request itself was generated by its 'forecast_entry'.

Properties: Export-Only Field

forecast_entry -- a Forecast_Entry field of model Delivery_Request

If nonexistent, then this is an actual (customer) Delivery_Request. Otherwise, this is the Forecast_Entry that generated this Delivery_Request.

Properties: Export-Only Field

delivery_promise -- a Delivery_Promise field of model Delivery_Request

The Delivery_Promise that has been made in response to this Delivery_Request.

Properties: Export-Only Field

item_requests -- a list of Item_Request submodels of model Delivery_Request
The individual order-line-items (Items) being requested.

due -- a Date_Range field of model Delivery_Request

The Date_Range within which the delivery should be made to the 'customer_plan'. Sometime between 'due_start' and 'due_end', the full Quantity specified in each of the item_requests should be satisfied.

If the 'due_end' Date is beyond the 'Plan horizon', (including the default "oo_future"), then this is considered to be a request for nothing. No Problem or Field_Error is created. In effect, this Delivery_Request is "turned off". This state is particularly useful while importing a set of requests.

Default: forever

promising_policy -- an extension selector of model Delivery_Request

This policy specifies the constraints on offering a Promise for the corresponding Request.

Default: ASAP

Extensions:

BUCKETED_ALL, BUCKETED_ASAP_ON_TIME_ALL, ALL_ON_TIME, ASAP, ASAP_MONTHLY, BUCKETED_ALLOCATION, BUCKETED_ALL_MIN_PRICE, BUCKETED_MIN_PRICE_ASAP, SHIP_IN_RATIO.

fulfillment_policy - - an extension selector of model Delivery_Request

This policy specifies restrictions on how a delivery is fulfilled.

Default: UNRESTRICTED

Extensions:

ON_TIME, FULL_QUANTITIES_OF_ALL_ITEMS, UNRESTRICTED.

max_price - - a Money field of model Delivery_Request

The maximum price desired by the customer_plan for this Delivery_Request. This is used as a guideline (not hard constraint) in order quoting.

Default: 00

seller_plan - - a Seller_Plan field of model Delivery_Request

The Seller that is responsible for this Delivery_Request. This may be the same as or a member of owner:seller_plan.

Default: owner:seller_plan

customer_plan - - a Site_Plan field of model Delivery_Request

The Site placing this Delivery_Request. This may be the same as or a member of owner:customer_plan.

Default: owner:customer_plan

name_from_customer - - a Symbol field of model Delivery_Request

The customer's name for this Delivery_Request. This may be the same as name or different. For example, this is often the customer's PO# (purchase order number). However, the supplier will often name Requests differently - after all, there is no guarantee that names (PO#s) will be unique, since they come from different customers.

Default: name

rank - - a Number field of model Delivery_Request

The customer-specified rank, weighting importance of this Delivery_Request relative to all other Delivery_Requests from this customer.

Note, as with all ranks, the higher the Number, the higher the rank, the greater the importance, the larger the weight.

Default: 0

delivery_name - - a String field of model Delivery_Request

The name of the customer to which this Delivery_Request should be delivered.

Default: owner:delivery_name

delivery_contact - - a String field of model Delivery_Request

The name of the individual to whom this Delivery_Request should be delivered.

Default: owner:delivery_contact

delivery_phone - - a String field of model Delivery_Request

The phone number of the delivery_contact to which this Delivery_Request should be delivered.

Default: owner:delivery_phone

delivery_fax - - a String field of model Delivery_Request

The fax phone number of the delivery_contact to which this Delivery_Request should be delivered.

Default: owner:delivery_fax

delivery_address - - a String field of model Delivery_Request

The street address to which this Delivery_Request should be delivered.

Default: owner:delivery_address

delivery_city - - a String field of model Delivery_Request

The city (township) to which this Delivery_Request should be delivered.

Default: owner:delivery_city

delivery_state - - a String field of model Delivery_Request

The state (province, territory) to which this Delivery_Request should be delivered.

Default: owner:delivery_state

delivery_country - - a String field of model Delivery_Request

The country (nation) to which this Delivery_Request should be delivered.

Default: owner:delivery_country

delivery_postal_code - - a String field of model Delivery_Request

The postal code (zip code) to which this Delivery_Request should be delivered.

Default: owner:delivery_postal_code

Models

Delivery_Request Model

promised_late - - *a Time field of model Delivery_Request*
If this Delivery_Request promises to deliver later than requested by 'item_request', then this returns how much later; otherwise it returns "0". This is infinite if 'unanswered' is "true". If this is non-zero and finite, then there is either a REQUEST_PROMISED_LATE Problem for this Delivery_Request. If infinite, then there is a DELIVERY_REQUEST_UNANSWERED Problem.

Properties: Export-Only Field

promised_early - - *a Time field of model Delivery_Request*
If this item_request promises to deliver earlier than requested by 'item_request', then this returns how much earlier; otherwise it returns "0". This is "0" if 'unanswered' is "true". If this is non-zero, then there is a REQUEST_PROMISED_EARLY Problem for this item_request.

Properties: Export-Only Field

promised_overpriced - - *a Money field of model Delivery_Request*
If this Delivery_Request offers a price that is more than the 'delivery_request.max_price', then this returns how much more; otherwise it returns "0". This is "0" if 'unanswered' is "true". If this is non-zero, then there is a REQUEST_PROMISED_OVERPRICED Problem for this item_request.

Properties: Export-Only Field

cancel - - *a Void field of model Delivery_Request*
This command cancels all the item_requests associated with this delivery_request. All the item_requests will be marked as cancelled but none of them will be deleted. All the associated delivery and receiving plans will be destroyed.

Properties: command=True Export-Only Field

plan_to_satisfy - - *a Void field of model Delivery_Request*
This command sets the plan for this Delivery_Request such that it is attempting to satisfy this Delivery_Request, and all of its item_requests. The 'delivery_request.delivery_plan.motive' will be set to match this Delivery_Request's 'due' and all its item_requests' quantities.

Alternatively, 'delivery_request.plan_to_satisfy' can be performed to set the plan to attempt to satisfy the promises for this request. Or, the 'item_request.delivery_plan.motive' can be set directly to attempt any subset of this Delivery_Request or its 'delivery_request'.

Properties: command=True Export-Only Field

Models

Delivery_Request Model

plan_as_requested - - *a Void field of model Delivery_Request*
This command sets the plan for this Delivery_Request such that it is attempting to satisfy this Delivery_Request, and all of its item_requests.

Properties: command=True Export-Only Field

not_planned - - *a Logical field of model Delivery_Request*
If "true", then no plan has been formed to meet at least parts of this Delivery_Request. This is "true" if 'due' is finite ('unanswered' is "false") and at least one of the 'item_request' is unplanned; if "true", then there is a REQUEST_NOT_PLANNED Problem for those unplanned item_promises.

Properties: Export-Only Field

planned_late - - *a Time field of model Delivery_Request*
Returns how late this Delivery_Request is currently planned to be fully delivered. It is the latest of each of the 'item_requests'. Note that 'delivery_request.planned_late' is equivalent to 'max(delivery_request.item_requests.for_each(#planned_late)'. If non-zero, then there is a REQUEST_PLANNED_LATE Problem for at least one of the 'item_requests'.

Unlike the individual 'item_requests', a Delivery_Request can have both 'planned_late' and 'planned_early' non-zero.

Properties: Export-Only Field

planned_early - - *a Logical field of model Delivery_Request*
Returns how early this Delivery_Request is currently planned to be delivered, at least in part. It is the earliest of each of the 'item_requests'. Note that 'delivery_request.planned_early' is equivalent to 'min(delivery_request.item_requests.for_each(#planned_early))'. If non-zero, then there is a REQUEST_PLANNED_EARLY Problem for at least one of the 'item_requests'.

Unlike the individual 'item_requests', a Delivery_Request can have both 'planned_late' and 'planned_early' non-zero.

Properties: Export-Only Field

last_change - - *a Date field of model Delivery_Request*
The Date at which the most recent change was made to this Delivery_Request. This is set by setting this or any other field in this model or the 'item_requests'. Note that setting this directly will set it to the specified Date, which may not be the current Date.

Model	Delivery_Request Model
-------	------------------------

This is used to support the changed_request functions of Slic, which provide net-change Exporting of Requests.

Default: now

problems, problems (Date_Range) , problems (Problem_Category) , problems (Date_Range, Problem_Category) -- a List(Problem)field of model

Delivery_Request

The Problems associated with this Delivery_Request, including any problems that are associated with the underlying Item_Request model. If passed a Date_Range, only the Problems whose dates overlap are returned. If passed a Problem_Category, only the Problems with that category are returned.

Properties: Export-Only Field

promised_date_problem -- a Problem field of model Delivery_Request

Obsolete! Use Delivery_Request.problems. The REQUEST_PROMISED_DATE Problem, if any, that exists between this Item_Promise and the Item_request it is answering. See fields unanswered, promised_late, and promised_early for alternate ways to test for REQUEST_UNANSWERED, REQUEST_PROMISED_LATE, and REQUEST_PROMISED_EARLY Problems, respectively.

Properties: obsolete=True Export-Only Field

promised_price_problem -- a Problem field of model Delivery_Request

Obsolete! Use Delivery_Request.problems. The REQUEST_PROMISED_OVERPRICED Problem, if any, that exists between this Item_Promise and the Item_request it is answering. See the field overpriced for an alternate way to test for this Problem.

Properties: obsolete=True Export-Only Field

plan_problems -- a List(Problem)field of model Delivery_Request

Obsolete! Use Delivery_Request.problems. A List of the REQUEST_PLAN Problems of the Item_requests; if any. See fields unplanned, planned_late, and planned_early for alternate ways to test for REQUEST_NOT_PLANNED, REQUEST_PLANNED_LATE, and REQUEST_PLANNED_EARLY Problems, respectively.

Note well: the lack of a Problem here only indicates that a plan to meet this Request is being attempted. There may be feasibility Problems with the plans associated with fulfilling this Request in addition to any Problem here.

Properties: obsolete=True Export-Only Field

Model	Delivery_Request Model
-------	------------------------

owner -- a Request field of model Delivery_Request

Properties: Export-Only Field

Models	Item_Request Model
--------	--------------------

5.1.3.1.7.1.1 Item_Request Model

Item_Request -- a submodel of model Delivery_Request

A Item_Request is a request for supply of a particular Item. This generally corresponds to the order "line item". Note that the due Date is specified by the 'owner' Delivery_Request, which allows several line items to be defined to be delivered together (same due Date_Range plus they should all be late if one is).

The Item is specified by the configuration field, which contains an 'item' field. The Configuration field supports configure-to-order products, and other non-STANDARD items.

The Item_Request model has fields that references these models : Configuration, Item, Item_Promise, Operation_Plan, Problem, Delivery_Request.

These models have a field that is a Item_Request model :

Delivery_Request, Item_Promise, REQUEST_NOT_PLANNED, REQUEST_PLANNED_LATE, REQUEST_PLANNED_EARLY, REQUEST_PLANNED_SHORT, REQUEST_PLANNED_EXCESS, PROMISE_NOT_PLANNED, PROMISE_PLANNED_LATE, PROMISE_PLANNED_EARLY, PROMISE_PLANNED_SHORT, PROMISE_PLANNED_EXCESS, ACCEPTANCE_NOT_PLANNED, ACCEPTANCE_PLANNED_LATE, ACCEPTANCE_PLANNED_EARLY, ACCEPTANCE_PLANNED_SHORT, ACCEPTANCE_PLANNED_EXCESS, DELIVER, REQUEST_PROMISED_SHORT, REQUEST_PROMISED_EXCESS, ITEM_PROMISE_OVERPRICED, SUPPLY_PLANNED_LATE, SUPPLY_PLANNED_EARLY, SUPPLY_PLANNED_SHORT, SUPPLY_PLANNED_EXCESS, SUPPLY_PROMISED_LATE, SUPPLY_PROMISED_EARLY, SUPPLY_PROMISED_SHORT, SUPPLY_PROMISED_EXCESS, REQUEST_FIXED, REQUEST_FIXED_WITH_ANALYSIS.

The key field for this model is name
This model may be extended with user-defined fields.

Models	Item_Request Model
--------	--------------------

name -- a Symbol field of model Item_Request

The name of this Item_Request. Depending upon the environment, this may be the order line-item number, it may be the name of the Item (Item family, or pseudo item), or otherwise. In the case of single Item orders, it may be the order-id or an empty String. It simply must be unique within the Delivery_Request, but otherwise can be whatever is meaningful.

Default: None -- this is a key field

configuration -- a Configuration field of model Item_Request

The Item being requested and the specific characteristics desired. The Item may be a STANDARD Item (no characteristics to specify), a Request-specific configuration of a OPTIONED Item, or a Request-specific CUSTOM Item (among others).

This is not settable itself, but you can set the fields of this Configuration.

If the Item is [unspecified] (the default), then this is considered to be a request for nothing. No Problem or Field_Error is created. In effect, this Item_Request is "turned off". Similarly, setting 'quantity' to zero (the default) or 'owner.due' to "oo_future" (the default) will effectively "turn off" this Item_Request. This state is particularly useful while importing a set of requests.

Properties: command=True Export-Only Field

item -- a Item field of model Item_Request

This is simply a shortcut to configuration.item.

If the Item is [unspecified] (the default), then this is considered to be a request for nothing. No Problem or Field_Error is created. In effect, this Item_Request is "turned off". Similarly, setting 'quantity' to zero (the default) or 'owner.due' to "oo_future" (the default) will effectively "turn off" this Item_Request. This state is particularly useful while importing a set of requests.

Default: [unspecified]

quantity -- a Quantity_Range field of model Item_Request

The range of Quantity of the configuration that is being requested as part of the 'owner' Delivery_Request.

Often this is a zero-interval range -- a particular Quantity is being requested. But in some industries, particular where the yield varies or the Items are specialized, giving a range from the minimum you need to the maximum that you will accept can prevent unnecessary waste, making the supply chain more economical.

These Quantity's are converted to the 'unit' of the Item being requested.

If the quantity is "0" (the default), then this is considered to be a request for nothing. No Problem or Field_Error is created. In effect, this Item_Request is "turned off". Similarly, setting item to [unspecified] (the default) or 'owner:due' to "oo_future" (the default) will effectively "turn off" this Item_Request. This state is particularly useful while importing a set of requests.

Default: 0

cancel - - a Void field of model Item_Request

This command marks this Item_Request as cancelled but does not destroy it. This command will destroy the associated delivery and receiving plans.

Properties: command=True Export-Only Field

cancelled - - a Logical field of model Item_Request

If "true", then this Item_Request has been cancelled. Otherwise, this Item_Request is active and should be considered when promising and planning.

Default: false

item_promise - - a Item_Promise field of model Item_Request

The Item_Promise that has been made in response to this Item_Request.

Properties: Export-Only Field

max_price - - a Money field of model Item_Request

The maximum price desired by the customer for this Item_Request. This is used as a guideline (not hard limit) in order quoting.

Default: oo

promised_short - - a Quantity field of model Item_Request

If this Item_Promise promises less than requested by this Item_Request, then this returns how much less; otherwise it returns "0". This is "0" if quantity is "0" or 'owner:due' is infinite. If this is non-zero, then there is a

REQUEST_PROMISED_SHORT Problem for this Item_Request.

Properties: Export-Only Field

promised_excess - - a Quantity field of model Item_Request

If this Item_Promise promises more than requested by this Item_Request, then this returns how much more; otherwise it returns "0". This is "0" if quantity is "0" or 'owner:due' is infinite. If this is non-zero, then there is a

REQUEST_PROMISED_EXCESS Problem for this Item_Request.

Properties: Export-Only Field

promised_overpriced - - a Money field of model Item_Request

If this Item_Promise offers a price that is more than the max_price, then this returns how much more; otherwise it returns "0". This is "0" if quantity is "0" or 'owner:due' is infinite. If this is non-zero, then there is a REQUEST_PROMISED_OVERPRICED Problem for this Item_Request.

Properties: Export-Only Field

delivery_plan - - a Operation_Plan field of model Item_Request

The Operation that has been planned to deliver this Item_Request. It is the same as item_promise.delivery_plan.

Properties: Export-Only Field

plan_to_satisfy - - a Void field of model Item_Request

This command sets the plan for this Item_Request such that it is attempting to satisfy this Request. The item_promise.delivery_plan motive will be set to match this Item_Request's quantity and its delivery_request's due Dates.

Alternatively, item_promise.plan_to_satisfy can be performed to set the plan to attempt to satisfy the promises for this request. Or, the item_promise.delivery_plan motive can be set directly to attempt any subset of this Item_Request or its Item_Promise.

Properties: command=True Export-Only Field

receiving_plan - - a Operation_Plan field of model Item_Request

The Operation that has been planned to receive this Item_Request.

Properties: Export-Only Field

plan_as_requested - - a Void field of model Item_Request

This command creates or re-sizes the receiving_plan for this Item_Request such that it satisfy this Request.

Properties: command=True Export-Only Field

not_planned - - a Logical field of model Item_Request

If "true", then no Operation has been planned to satisfy this Item_Request. This is "false" if 'owner:unanswered' is "true". If "true", then there is a REQUEST_NOT_PLANNED Problem for this Item_Request.

Properties: Export-Only Field

Models

Item_Request Model

planned_late - - a Time field of model Item_Request

If the plan for this Item_Request is planned for later than requested, then this returns how much later; otherwise it returns "0". This is "0" if quantity is "0" or ownerdue is infinite. If this is non-zero, then there is a REQUEST_PLANNED_LATE Problem for this Item_Request.

Properties: Export-Only Field

planned_early - - a Time field of model Item_Request

If the plan for this Item_Request is planned for earlier than requested, then this returns how much earlier; otherwise it returns "0". This is "0" if quantity is "0" or ownerdue is infinite. If this is non-zero, then there is a REQUEST_PLANNED_EARLY Problem for this Item_Request.

Properties: Export-Only Field

planned_short - - a Quantity field of model Item_Request

If the plan for this Item_Request is planned to deliver less than requested, then this returns how much less; otherwise it returns "0". This is "0" if quantity is "0" or ownerdue is infinite. If this is non-zero, then there is a REQUEST_PLANNED_SHORT Problem for this Item_Request.

Properties: Export-Only Field

planned_excess - - a Quantity field of model Item_Request

If the plan for this Item_Request is planned to deliver more than requested, then this returns how much more; otherwise it returns "0". This is "0" if quantity is "0" or ownerdue is infinite. If this is non-zero, then there is a REQUEST_PLANNED_EXCESS Problem for this Item_Request.

Properties: Export-Only Field

last_change - - a Date field of model Item_Request

The Date at which the most recent change was made to this Item_Request. This is set by setting this or any other field in this model. Note that setting this directly will set it to the specified Date, which may not be the current Date.

This is used to support the 'changed_requests' functions of Slic, which provide net-change Exporting of Requests.

This is the same as calling owner.r_last_change.

Default: now

Models

Item_Request Model

problems, problems (Date, Range) , problems (Problem, Category) , problems (Date, Range, Problem, Category) - - a List(Problem) field of model

Item_Request

The Problems associated with this Item_Request. If passed a Date, Range, only the Problems whose 'dates' overlap are returned. If passed a Problem, Category, only the Problems with that 'category' are returned.

Properties: Export-Only Field

promised_quantity_problem - - a Problem field of model Item_Request

Obsolete! Use Item_Request.problems! The REQUEST_PROMISED_QUANTITY Problem, if any, that exists between this Item_Request and the Item_promise, answering it. See fields promised_short and promised_excess for more convenient ways to test for REQUEST_PROMISED_SHORT, and REQUEST_PROMISED_EXCESS Problems, respectively. This will be nonexistent if there is no such Problem or if 'owner:unanswered' is "true".

Properties: obsolete=True Export-Only Field

promised_price_problem - - a Problem field of model Item_Request

Obsolete! Use Item_Request.problems! The REQUEST_PROMISED_OVERPRICED Problem, if any, that exists between this Item_Request and the Item_promise, answering it. See the field 'overpriced' for an alternate way to test for this Problem. This will be nonexistent if there is no such Problem or if 'owner:unanswered' is "true".

Properties: obsolete=True Export-Only Field

planned_date_problem - - a Problem field of model Item_Request

Obsolete! Use Item_Request.problems! The REQUEST_PLANNED_DATE Problem, if any, that exists between this Item_Request and the Operation_Plan currently planned to satisfy it (delivery_plan). See fields 'unplanned', 'planned_late', and 'planned_early' for alternate ways to test for REQUEST_NOT_PLANNED, REQUEST_PLANNED_LATE, and REQUEST_PLANNED_EARLY Problems, respectively.

Note well: the lack of a Problem here only indicates that a plan to meet this Request is being attempted. There may be feasibility Problems with the plans associated with fulfilling this Promise in addition to any Problem here.

Properties: obsolete=True Export-Only Field

planned quantity, problem - - a Problem field of model Item, Request
 Obsolete! Use Item_Request problems! The REQUEST_PLANNED_QUANTITY
 Problem, if any, that exists between this Item_Promise and the Operation_Plan cur-
 rently planned to satisfy it (delivery_plan). See fields 'unplanned', 'planned_short',
 and 'planned_excess' for more convenient ways to test for
 REQUEST_NOT_PLANNED, REQUEST_PLANNED_SHORT, and
 REQUEST_PLANNED_EXCESS Problems, respectively.

Note well: the lack of a Problem here only indicates that a plan to meet this Request is
 being attempted. There may be feasibility Problems with the plans associated with
 fulfilling this Promise in addition to any Problem here.

Properties: obsolete=True Export-Only Field

owner - - a Delivery_Request field of model Item_Request

Properties: Export-Only Field

5.1.3.1.8 Promise Model

Promise - - a submodel of model Site_Plan

A Promise model is the 'owner' supplier's response to a customer's Request. Note that a
 Promise model exists for every Request model. However, the Promise may not be
 'offered' yet; or may be offered, but offering zero Quantity or to be delivered by the
 infinite future (Request has been rejected). The Promise model is either of those
 responses, or the lack of an offer, to the Request.

Once offered, a Promise model is a commitment from the 'owner' to supply a set of
 deliveries, each with a set of Items. It also becomes, once accepted, a commitment
 from the 'customer' to purchase the supplied Items.

A Promise can be offered to actual Requests from customers, or to forecasted
 Requests. In either case, the commitment from the supplier is real. Promises to fore-
 cast Requests can be 'consumed' in order to make Promises for actual Requests. Such
 consumption does not require the involvement of the supplier. Thus, Promises made
 by the 'supplier' for forecast Requests that have not yet been consumed by actual
 Requests represent ATP (Available-To-Promise). As such, Promises for forecast
 Requests are as binding as Promises for actual Requests.

For a Promise to be offered for a Request, a Plan may be created that can satisfy each
 of the Item_Requests within each of the Delivery_Requests of the Request. If no Plan
 can be found that fully satisfies the Request, a lesser Promise may still be offered. In
 that case, the Promise should detail what is being offered corresponding to each ele-
 ment of the Request. For each Item_Request is needed an Item_Promise, and for each
 Delivery_Request a Delivery_Promise. Thus, the Promise models form a parallel hier-
 archy to the Request models they are promising.

For the case of Requests that model contracts or "blanket" orders, the exact
 Delivery_Requests will not have been generated yet. Rather, the Promise commits to
 delivering any Delivery_Requests that can be generated within the bounds specified by
 the 'delivery_policy'. To model and plan for that, the 'delivery_policy' will generate
 forecasted Delivery_Requests that mimic the actual Delivery_Requests that are
 allowed. Plans will be made to satisfy those 'estimated' Delivery_Requests, and the
 Promise will be based upon those Plans.

For more detail on Requests, the hierarchy, or the normal Request-Promise protocol,
 please see the Request model documentation.

The model has selectors:

delivery_policy.

The Promise model has these submodels:

Delivery_Promise.

The Promise model has fields that reference these models:

Request, Delivery_Promise, Acceptance, Site_Plan.

These models have a field that is a Promise model:

LINK, Request, Acceptance, SUPPLIER, Delivery_Promise,
 REQUEST_NOT_PLANNED, REQUEST_PLANNED_LATE,
 REQUEST_PLANNED_EARLY, REQUEST_PLANNED_SHORT,
 REQUEST_PLANNED_EXCESS, PROMISE_NOT_PLANNED,
 PROMISE_PLANNED_LATE, PROMISE_PLANNED_EARLY,
 PROMISE_PLANNED_SHORT, PROMISE_PLANNED_EXCESS,
 ACCEPTANCE_NOT_PLANNED, ACCEPTANCE_PLANNED_LATE,
 ACCEPTANCE_PLANNED_EARLY, ACCEPTANCE_PLANNED_SHORT,
 REQUEST_PROMISED_EARLY, REQUEST_PROMISED_SHORT,
 REQUEST_PROMISED_EXCESS, ITEM_PROMISE_OVERPRICED,
 DELIVERY_PROMISE_OVERPRICED, PROMISE_NOT_OFFERED,
 PROMISE_NOT_CONFIRMED, PROMISE_NOT_ACCEPTED,

ACCEPTANCE_INCONSISTENT, REQUEST_QUEUED, DELIVERY_REQUEST_NOT_COORDINATED, DELIVERY_PROMISE_NOT_COORDINATED, DELIVERY_ACCEPTANCE_NOT_COORDINATED, SUPPLY_PLANNED_LATE, SUPPLY_PLANNED_EARLY, SUPPLY_PLANNED_SHORT, SUPPLY_PLANNED_EXCESS, SUPPLY_PROMISED_LATE, SUPPLY_PROMISED_EARLY, SUPPLY_PROMISED_SHORT, SUPPLY_PROMISED_EXCESS.

The key field for this model is request.
This model may be extended with user-defined fields.

request - - *a Request field of model Promise*
The Request to which this Promise responds.
Default: None - - this is a key field
Properties: Export-Only Field

delivery_promises - - *a list of Delivery_Promise submodels of model Promise*
The individual promises for delivery_requests.

delivery_policy - - *an extension selector of model Promise*
Defines how 'delivery_promises' may be adjusted, without "changing" the promise. This is typically the same as 'request.delivery_policy', but need not be. The 'owner' may promise a different policy than the 'customer' requested. Of course, the 'customer' need not accept the Promise.
Default: FIXED
Extensions:

FIXED.

acceptance - - *a Acceptance field of model Promise*
The acceptance in response to this promise
Properties: Export-Only Field

accept_by - - *a Date field of model Promise*
Once offered, this Promise is just an offer until accepted by the 'customer'. The offer is good until this 'accept_by' Date. The 'customer' must accept this Promise by this Date or the Promise will expire.
Default: request.accept_by

date_offered - - *a Date field of model Promise*
The Date that the supplier promised delivery to the 'customer'. Note that the Promise is for the 'delivery_promises' and 'delivery_policy' in this Promise, which may not be exactly what was requested by the Request.

If this Date is prior to 'request.date_issued', then this Promise has not been offered. If this Date is on or after 'request.date_issued', then this Promise has been offered as the answer to the 'request', and the 'request' will no longer be considered (no longer raise PROMISE_NOT_OFFERED or REQUEST_NOT_PLANNED Problems).
Default: oo_past

plan_to_satisfy - - *a Void field of model Promise*
This command creates plans to satisfy this Promise. It does this by simply calling 'plan_to_satisfy' on each of its 'delivery_promises'. Thus, it is equivalent to 'delivery_promises.for_each(#,plan_to_satisfy)'.

Alternatively, 'request.plan_to_satisfy' can be performed to call 'plan_to_satisfy' on each of the 'delivery_requests'.
Properties: command=True Export-Only Field

promise_as_planned, promise_as_planned (Logical) - - *a Void field of model Promise*
This command sets this Promise to promise what has been planned for this Promise. It does this by simply calling 'promise_as_planned' on each of its 'delivery_promises'. Thus, it is equivalent to 'delivery_promises.for_each(#,promise_as_planned)'.
Properties: command=True Export-Only Field

plan_as_promised - - *a Void field of model Promise*
This command sets the plan for this 'Promise' such that it is attempting to satisfy this Promise, and all of its 'delivery_promises' and 'item_promises'.
Properties: command=True Export-Only Field

reject - - *a Void field of model Promise*
This command zeros all Item_Promises within this Promise and sets 'offered' to 'now'.
Properties: command=True Export-Only Field

last_change - - *a Date field of model Promise*
The Date at which the most recent change was made to this Promise. This is set by setting this or any other field in this model or the 'delivery_promises'. Note that setting this directly will set it to the specified Date, which may not be the current Date.

Models	Promise Model
--------	---------------

This is used to support the 'changed_promises' functions of Site, which provide net-change Exporting of Promises.
Default: now

problems, problems (Date_Range) , problems (Problem_Category) , problems (Date_Range, Problem_Category) - - a List(Problem) field of model Promise
The Problems associated with this Promise, including any problems that are associated with the underlying Delivery_Promise and Item_Promise models. If passed a Date_Range, only the Problems whose dates overlap are returned. If passed a Problem_Category, only the Problems with that category are returned.
Properties: Export-Only Field

offered - - a Date field of model Promise
Obsolete! This field has been renamed 'date_offered' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.
Default: oo_past

Properties: obsolete=True

owner - - a Site_Plan field of model Promise

Properties: Export-Only Field

Models	Delivery_Promise Model
--------	------------------------

5.1.3.1.8.1 Delivery_Promise Model

Delivery_Promise - - a submodel of model Promise

Delivery_Promise models the supplier's response to a 'customer's Delivery_Request for a set of Items to be delivered together. Several Delivery_Promises may make up the 'owner' Promise.

Note that a Delivery_Promise model exists for every Delivery_Request model. However, the Delivery_Promise may not be 'promised'; in fact, it may even be 'rejected'. The Delivery_Promise models either of those responses, and the lack of a response, to the Delivery_Request.

A Delivery_Promise can be a response to 'actual' Delivery_Requests from customers, or can be a response to 'forecasted' or 'generated' Delivery_Requests. In either case, any commitment from the supplier is real. As 'actual' Delivery_Requests consume from 'forecast' Delivery_Requests, the Delivery_Promises corresponding to the 'forecast' Delivery_Requests are converted into Delivery_Promises for the 'actual' Delivery_Requests. Such conversion does not require the involvement of the supplier. Thus, Delivery_Promises made by the 'supplier' for 'forecast' Delivery_Requests that have not yet been consumed by 'actual' Delivery_Requests represent ATP (Available-To-Promise). As such, Delivery_Promises for 'forecast' Delivery_Requests are as binding as Delivery_Promises for 'actual' Delivery_Requests.

The model has selectors:
fulfillment_policy.

The Delivery_Promise model has these submodels :
Delivery_Available_To_Promise, Item_Promise.

The Delivery_Promise model has fields that reference these models :
Delivery_Request, Delivery_Acceptance, Delivery_Available_To_Promise, Item_Promise, Problem.

These models have a field that is a Delivery_Promise model :
Promise, Delivery_Request, Delivery_Acceptance, Delivery_Available_To_Promise, Item_Promise, REQUEST_NOT_PLANNED, REQUEST_PLANNED_LATE, REQUEST_PLANNED_EARLY, REQUEST_PLANNED_SHORT, REQUEST_PLANNED_EXCESS, PROMISE_NOT_PLANNED, PROMISE_PLANNED_LATE, PROMISE_PLANNED_EARLY, PROMISE_PLANNED_SHORT, PROMISE_PLANNED_EXCESS,

ACCEPTANCE_NOT_PLANNED, ACCEPTANCE_PLANNED_LATE, ACCEPTANCE_PLANNED_EARLY, ACCEPTANCE_PLANNED_SHORT, ACCEPTANCE_PLANNED_EXCESS, REQUEST_PROMISED_LATE, REQUEST_PROMISED_EARLY, REQUEST_PROMISED_SHORT, REQUEST_PROMISED_EXCESS, ITEM_PROMISE_OVERPRICED, DELIVERY_PROMISE_OVERPRICED, PROMISE_NOT_CONFIRMED, DELIVERY_REQUEST_NOT_COORDINATED, DELIVERY_PROMISE_NOT_COORDINATED, DELIVERY_ACCEPTANCE_NOT_COORDINATED, SUPPLY_PLANNED_LATE, SUPPLY_PLANNED_EARLY, SUPPLY_PLANNED_SHORT, SUPPLY_PLANNED_EXCESS, SUPPLY_PLANNED_LATE, SUPPLY_PROMISED_EARLY, SUPPLY_PROMISED_SHORT, SUPPLY_PROMISED_EXCESS.

The key field for this model is `delivery_request`.

This model may be extended with user-defined fields.

`delivery_request` -- *a Delivery_Request field of model Delivery_Promise*

The `Delivery_Request` to which this `Delivery_Promise` responds.

Default: None -- this is a key field

Properties: Export-Only Field

`delivery_acceptance` -- *a Delivery_Acceptance field of model Delivery_Promise*

The `Delivery_Acceptance` for this `Delivery_Promise`.

Properties: Export-Only Field

`delivery_atp` -- *a list of Delivery_Available_To_Promise submodels of model*

`Delivery_Promise`

The list of options for promising this `Delivery_Promise` for this Seller. For each `Item_Promise`, the list of `Item_Available_To_Promise` is computed and then sorted by delivery date. The result of this process is a list of delivery dates and what can be delivered for each `Item_Request` on that delivery date. This list can then be sorted by date at which time the first `Delivery_Available_To_Promise` will indicate the earliest date at which a delivery (it could be just a partial delivery) can be made. The last entry in the sorted list will indicate the earliest date on which the entire delivery can be made.

`promising_policy_atp` -- *a List(Delivery_Available_To_Promise) field of model*

`Delivery_Promise`

Return the quote (subset of `delivery_atp`) according to the promising policy in effect for the Request (see `Promising_Policy` extension of `Delivery_Request`).

Properties: Export-Only Field

`item_promises` -- *a list of Item_Promise submodels of model Delivery_Promise*

Promises for the individual order-line-items.

`due` -- *a Date_Range field of model Delivery_Promise*

The `Date_Range` within which the delivery is promised to be made to the customer. Sometime between 'due.start' and 'due.end', the full Quantity specified in each of the `Item_request` should be satisfied.

Default: forever

`rank` -- *a Number field of model Delivery_Promise*

The supplier-specified rank, weighing importance of this `Delivery_Promise` relative to all other `Delivery_Promise` owned by this supplier.

Note, as with all rank's, the higher the Number, the higher the rank, the greater the importance, the larger the weight.

Default: 0

`fulfillment_policy` -- *an extension selector of model Delivery_Promise*

This policy defines whether a delivery can be shorted or delayed.

Typically this is the same as 'delivery_request.fulfillment_policy'. However, if the supplier is not willing to do the requested fulfillment_policy, the promised one may be different.

Default: delivery_request.fulfillment_policy

Extensions:

ON_TIME, FULL_QUANTITIES_OF_ALL_ITEMS, UNRESTRICTED.

`date_confirmed` -- *a Date field of model Delivery_Promise*

The Date the Site confirmed this `Delivery_Promise` made by the Seller. When a Seller makes a `Promise`, it is generally based upon the availability of promises to forecast requests. Due to variance in how the forecast is modeled and reality, the Seller's `Promise` may not be feasible.

When a Site actually forms a plan to satisfy the promise and calls 'promise_as_planned' or 'confirm_planned_promises', this field is set to how 'to indicate the `Promise` has been confirmed.

Default: oo_past

`list_price` -- *a Money field of model Delivery_Promise*

The sum of 'list_price's for all the `Item_promises`. This is not settable here -- it is computed from the `Item_promises`.

Models

Delivery_Promise Model

Properties: Export-Only Field

sum_discount -- *a Percentage field of model Delivery_Promise*

The cumulative percentage discount from 'list_price', computed from the 'item_promises'. This is not scalable here -- it is computed from the 'item_promises'.

Default: 0%

Properties: Export-Only Field

sum_price -- *a Money field of model Delivery_Promise*

The sum of the prices of the 'item_promises'. This is not scalable here -- it is computed from the 'item_promises'.

Default: list_price

Properties: Export-Only Field

delivery_discount -- *a Percentage field of model Delivery_Promise*

The additional percentage discounted from 'sum_price' to get the quoted 'delivery_price'.

Default: 0%

delivery_price -- *a Money field of model Delivery_Promise*

The price being quoted for the full delivery as specified.

Default: sum_price

Properties: Export-Only Field

promised_late -- *a Time field of model Delivery_Promise*

If this Delivery_Promise promises to deliver later than requested by 'item_request', then this returns how much later; otherwise it returns '0'. This is infinite if 'unsatisfied' is 'true'. If this is non-zero and finite, then there is either a REQUEST_PROMISED_LATE Problem for this Delivery_Promise. If infinite, then there is a DELIVERY_REQUEST_UNANSWERED Problem.

Properties: Export-Only Field

promised_early -- *a Time field of model Delivery_Promise*

If this item_Promise promises to deliver earlier than requested by 'item_request', then this returns how much earlier; otherwise it returns '0'. This is '0' if 'unsatisfied' is 'true'. If this is non-zero, then there is a REQUEST_PROMISED_EARLY Problem for this item_Promise.

Properties: Export-Only Field

Models

Delivery_Promise Model

promised_overpriced -- *a Money field of model Delivery_Promise*

If this Delivery_Promise offers a price that is more than the 'delivery_request_max_price', then this returns how much more; otherwise it returns '0'. This is '0' if 'unsatisfied' is 'true'. If this is non-zero, then there is a REQUEST_PROMISED_OVERPRICED Problem for this item_Promise.

Properties: Export-Only Field

plan_to_satisfy -- *a Void field of model Delivery_Promise*

This command sets the plan for each of this Delivery_Promise's 'item_promises', such that it is attempting to satisfy this promise. It is equivalent to 'item_promises.for_each(plan_to_satisfy)'. The 'delivery_plan.motive' will be set to match this Delivery_Promise's 'due Dates' and its 'item_promises' quantity's.

Alternatively, 'delivery_request.plan_to_satisfy' can be performed to set the plan to attempt to satisfy the request for this promise. Or, the 'delivery_plan.motive' can be set directly to attempt any subset of this Delivery_Promise or its 'delivery_request'.

Properties: command=True Export-Only Field

plan_as_promised -- *a Void field of model Delivery_Promise*

This command sets the plan for this Delivery_Promise such that it is attempting to satisfy this Delivery_Promise, and all of its 'item_promises'.

Properties: command=True Export-Only Field

promise_as_planned -- *a Void field of model Delivery_Promise*

This command sets this Delivery_Promise and its 'item_promises' according to what is currently planned (and what was requested by 'delivery_request').

The 'due Date Range' will be set to include 'delivery_request.due' and the planned Date. (The 'due.start' is set to the earlier of the 'delivery_request.due.start' and the planned Date. The 'due.end' is set to the later of the 'delivery_request.due.end' and the planned Date.)

Each of the 'item_promises' quantity's are set similarly, based upon the planned Quantity's.

Properties: command=True Export-Only Field

promise_by_policy -- *a Void field of model Delivery_Promise*

This command sets this Delivery_Promise and its 'item_promises' according to 'Delivery_Available_To_Promise' and the promising_policy in effect for the Delivery_Request.

Properties: command=True Export-Only Field

not_planned - - *a Logical field of model Delivery_Promise*

If "true", then no plan has been formed to meet at least parts of this Delivery_Promise. This is "true" if 'due' is finite (unanswered) is "false") and at least one of the item_promises is unplanned. If "true", then there is a PROMISE_NOT_PLANNED Problem for those unplanned item_promises.

Properties: Export-Only Field

planned_late - - *a Time field of model Delivery_Promise*

Returns how late this Delivery_Promise is currently planned to be fully delivered. It is the latest of each of the item_promises. Note that delivery_promise_planned_late is equivalent to max(delivery_promise.item_promises.for_each{#,planned_late}). If non-zero, then there is a PROMISE_PLANNED_LATE Problem for at least one of the item_promises.

Unlike the individual item_promises, a Delivery_Promise can have both planned_late and planned_early non-zero.

Properties: Export-Only Field

planned_early - - *a Logical field of model Delivery_Promise*

Returns how early this Delivery_Promise is currently planned to be delivered, at least in part. It is the earliest of each of the item_promises. Note that delivery_promise_planned_early is equivalent to min(delivery_promise.item_promises.for_each{#,planned_early}). If non-zero, then there is a PROMISE_PLANNED_EARLY Problem for at least one of the item_promises.

Unlike the individual item_promises, a Delivery_Promise can have both planned_late and planned_early non-zero.

Properties: Export-Only Field

last_change - - *a Date field of model Delivery_Promise*

The Date at which the most recent change was made to this Delivery_Promise. This is set by setting this or any other field in this model or the item_promises. Note that setting this directly will set it to the specified Date, which may not be the current Date.

This is used to support the 'changed_promises' functions of Site, which provide net-change Exporting of Promises.

Default: now

problems, **problems (Date_Range)**, **problems (Problem_Category)**, **problems (Date_Range, Problem_Category)** - - *a List(Problem) field of model Delivery_Promise*

The Problems associated with this Delivery_Promise, including any problems that are associated with the underlying item_Promise model. If passed a Date_Range, only the Problems whose 'dates' overlap are returned. If passed a Problem_Category, only the Problems with that 'category' are returned.

Properties: Export-Only Field

confirmed - - *a Date field of model Delivery_Promise*

Obsolete! This field has been renamed 'date_confirmed' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.

Default: oo_past

Properties: obsolete=True

promised_date_problem - - *a Problem field of model Delivery_Promise*

Obsolete! Use Delivery_Promise_problems. The REQUEST_PROMISED_DATE Problem, if any, that exists between this item_Promise and the item_request; it is answering. See fields 'unanswered', 'promised_late', and 'promised_early' for alternate ways to test for REQUEST_UNANSWERED, REQUEST_PROMISED_LATE, and REQUEST_PROMISED_EARLY Problems, respectively.

Properties: obsolete=True Export-Only Field

promised_price_problem - - *a Problem field of model Delivery_Promise*

Obsolete! Use Delivery_Promise_problems. The REQUEST_PROMISED_OVERPRICED Problem, if any, that exists between this item_Promise and the item_request; it is answering. See the field 'overpriced' for an alternate way to test for this Problem.

Properties: obsolete=True Export-Only Field

plan_problems - - *a List(Problem) field of model Delivery_Promise*

Obsolete! Use Delivery_Promise_problems. A List of the PROMISE_PLAN Problems of the item_promises, if any. See fields 'unplanned', 'planned_late', and 'planned_early' for alternate ways to test for PROMISE_NOT_PLANNED, PROMISE_PLANNED_LATE, and PROMISE_PLANNED_EARLY Problems, respectively.

Note well: the lack of a Problem here only indicates that a plan to meet this Promise is being attempted. There may be feasibility Problems with the plans associated with fulfilling this Promise in addition to any Problem here.

Properties: obsolete=True Export-Only Field

owner -- a Promise field of model Delivery_Promise

Properties: Export-Only Field

51.3.1.8.1.1 Delivery_Available_To_Promise Model

Delivery_Available_To_Promise -- a submodel of model Delivery_Promise

A Delivery_Available_To_Promise models a forecast Promise that is suitable for the 'owner:delivery_request', that has been allocated for use by the Seller, but has not yet been consumed by an actual (customer) Request. Such a forecast Promise is then "available" for being consumed to form a Promise for the 'owner:Delivery_Request'. This Delivery_Available_To_Promise takes the form of a delivery date and a list of Item_Available_To_Promises which constitute the available promises for each Item_Request in 'owner:delivery_request'.

Note that there may be numerous Products that match the 'owner:delivery_request's Item_Request. Given that, there may be numerous different forecast Promises that are available to Promise the 'owner:delivery_request's Item_Request. They may vary in Quantity, delivery Date, Configuration, and price. See the Product model for more detail on how an Item_Request is matched with a Product.

Furthermore, if the available allocation to the Seller for a particular Product is inadequate, the Seller can look at its organization's allocations for that Product.

Note that the "availability" recorded by these models can be fleeting. If more than one User has access to the same Seller allocations (particularly for organization's), then allocations can be consumed between the time that the Delivery_Available_To_Promise is computed and the decision is made to consume it. Or plan revisions may be made in that same time, causing allocations to disappear. How stable ATP computations are depends upon the environment and usage. By allocating to individual Users and by insulating the order quoting environment from plan changes, the ATP computations can be made very stable.

The Delivery_Available_To_Promise model has fields that references these models : Delivery_Promise.

These models have a field that is a Delivery_Available_To_Promise model : Delivery_Promise.

The key field for this model is delivery_date

delivery_date -- a Date field of model Delivery_Available_To_Promise
The delivery_date that could be Promised with this Delivery_Available_To_Promise.
Default: None -- this is a key field
Properties: Export-Only Field

item_atp ... a List(Item_Available_To_Promise) field of model Delivery_Available_To_Promise

The list of possibilities for promising owner.item_promises on 'delivery_date' for this Seller. There will be at most one Item_Available_To_Promise for each owner.item_promises to specify the ATP for that particular Item Request on 'delivery_date'. The absence of an Item_Available_To_Promise indicates that none of the Item Request can be delivered on 'delivery_date'.

Properties: Export-Only Field

offer_promise ... a Void field of model Delivery_Available_To_Promise

Consume this Delivery_Available_To_Promise in order to form a Promise for 'owner'. If this Delivery_Available_To_Promise has inadequate Quantity for any of its Item_Promises, then this will result in a new Delivery_Request with this Delivery_Available_To_Promise's Quantities. The appropriate owner.delivery_requests item_requests will be reduced in Quantity by like amount.

If the dates of this satisfy the 'owner.delivery_request', then the new Item_Request(s) will have the same Delivery_Request 'owner', and thus will be delivered together. However, if dates does not satisfy it, then a new Delivery_Request will be created as well.

Properties: command=True Export-Only Field

owner ... a Delivery_Promise field of model Delivery_Available_To_Promise

Properties: Export-Only Field

5.1.3.1.8.1.2 Item_Promise Model

Item_Promise ... a submodel of model Delivery_Promise

An Item_Promise is an agreement to supply/purchase a Quantity of a particular Item. The Date_Range within which this Quantity should be supplied is given by the 'owner' Delivery_Promise, which can also coordinate multiple Item_Promises together.

The Item_Promise model has these submodels :
Item_Available_To_Promise.

The Item_Promise model has fields that references these models :

Item_Request, Item_Acceptance, Item_Available_To_Promise, Forecast, Configuration, Operation_Plan, Problem, Delivery_Promise.

These models have a field that is a Item_Promise model :

Delivery_Promise, Item_Request, Item_Acceptance,
Item_Available_To_Promise, REQUEST_NOT_PLANNED,
REQUEST_PLANNED_LATE, REQUEST_PLANNED_EARLY,
REQUEST_PLANNED_SHORT, REQUEST_PLANNED_EXCESS,
PROMISE_NOT_PLANNED, PROMISE_PLANNED_LATE,
PROMISE_PLANNED_EARLY, PROMISE_PLANNED_SHORT,
PROMISE_PLANNED_EXCESS, ACCEPTANCE_NOT_PLANNED,
ACCEPTANCE_PLANNED_LATE, ACCEPTANCE_PLANNED_EARLY,
ACCEPTANCE_PLANNED_SHORT,
ACCEPTANCE_PLANNED_EXCESS, DELIVER,
REQUEST_PROMISED_SHORT, REQUEST_PROMISED_EXCESS,
ITEM_PROMISE_OVERRICED, SUPPLY_PLANNED_LATE,
SUPPLY_PLANNED_EARLY, SUPPLY_PLANNED_SHORT,
SUPPLY_PLANNED_EXCESS, SUPPLY_PROMISED_LATE,
SUPPLY_PROMISED_EARLY, SUPPLY_PROMISED_SHORT,
SUPPLY_PROMISED_EXCESS.

The key field for this model is item_request

Item_request ... a Item_Request field of model Item_Promise

The Item_Request that this Item_Promise is trying to satisfy.

Default: None -- this is a key field

Properties: Export-Only Field

Item_acceptance ... a Item_Acceptance field of model Item_Promise

The Item_Acceptance for this Item_Promise.

Models	Item_Promise Model
--------	--------------------

Properties: Export-Only Field

item_atp -- *a list of Item_Available_To_Promise submodels of model Item_Promise*

The list of options for promising from Item_Available_To_Promise for this Seller. For each Forecast in matching_forecasts, Item ATP is sought that will satisfy the item_request -- that will be put in one Item_Available_To_Promise model. If there is not enough Item ATP to cover the request on-time, then additional Item_Available_To_Promise models will be created for the next available Dates, until the max Quantity is covered, or there is no more Item ATP.

That is repeated for each Forecast in matching_forecasts so that all options that are available to promise are listed. In that way, the best option considering price, timing, delivery_lead_time, and configuration can be selected.

matching_forecasts -- *a List(Forecast) field of model Item_Promise*

A List of the Forecasts that could be consumed to match item_request. Only 'active' Forecasts are considered to match (see model Forecast). The match considers four criteria: 1. requested Item must be sold as Product(Root),

2. the requesting customer site must match the Product(Root) sites, 3. the requested quantity must be greater than the Product(Root) min_quantity, 4. the requested date must not be later than allowed by the Product(Root) min_lead_time, and 5. the request date must be within the effective_dates of the Product(Root).

Properties: Export-Only Field

consumed_forecast -- *a Forecast field of model Item_Promise*

The Forecast, if any, that is consumed by this Item_Promise. If nonexistent, then either this is itself a forecast Request, or this was an unexpected, unforecasted Request. The consumed_forecast may not be a GROUP Forecast.

Default: none

configuration -- *a Configuration field of model Item_Promise*

The item being requested and the specific characteristics desired. The Item may be a STANDARD Item (no characteristics to specify), a Request-specific configuration of a OPTIONED Item, or a Request-specific CUSTOM Item (among others).

This is almost always the same as 'item_request.configuration' (unless the requested configuration is not available or feasible). This is not settable itself, but you can set the fields of this Configuration.

Properties: command=True Export-Only Field

Models	Item_Promise Model
--------	--------------------

item -- *a Item field of model Item_Promise*
This is simply a shortcut to configuration.item.

If the 'item' is [unspecified] (the default), then this is considered to be a promise for nothing.

Default: [unspecified]

quantity -- *a Quantity_Range field of model Item_Promise*

The range of Quantity of the configuration that is being promised as part of the 'owner' Delivery_Promise. This is typically 'item_request.quantity'. However, if that Quantity was not feasible, then this Quantity may be less.

These Quantity's are converted to the unit of the 'item' being requested.
Default: 0

list_price -- *a Money field of model Item_Promise*

The standard price to be charged for the argument Item_Request, based on its Configuration, its quantity, its due Date, its confirm_by Date, and the delivery Location. This is used as a guideline in product quoting.

Default: 0.0

Properties: Export-Only Field

discount -- *a Percentage field of model Item_Promise*

The percentage discount from list_price. Setting this sets price. Setting price also sets this.

Default: 0%

price -- *a Money field of model Item_Promise*

The price being quoted to deliver as specified.

Default: list_price

promised_short -- *a Quantity field of model Item_Promise*

If this Item_Promise promises less than requested by item_request, then this returns how much less; otherwise it returns '0'. This is '0' if quantity is '0' or 'owner.due' is infinite. If this is non-zero, then there is a REQUEST_PROMISED_SHORT Problem for this Item_Promise.

Properties: Export-Only Field

Models**Item_Promise Model**

promised_excess - - *a Quantity field of model Item_Promise*

If this Item_Promise promises more than requested by Item_request, then this returns how much more; otherwise it returns "0". This is "0" if quantity is "0" or 'ownerdue' is infinite. If this is non-zero, then there is a REQUEST_PROMISED_EXCESS Problem for this Item_Promise.

Properties: Export-Only Field

promised_overpriced - - *a Money field of model Item_Promise*

If this Item_Promise offers a price that is more than the Item_request.max_price, then this returns how much more; otherwise it returns "0". This is "0" if quantity is "0" or 'ownerdue' is infinite. If this is non-zero, then there is a REQUEST_PROMISED_OVERPRICED Problem for this Item_Promise.

Properties: Export-Only Field

delivery_plan - - *a Operation, Plan field of model Item_Promise*

The Operation that has been planned to deliver this Item_Promise. It is the same as Item_request.delivery_plan.

Properties: Export-Only Field

plan_to_satisfy - - *a Void field of model Item_Promise*

This command sets the plan for this Item_Promise such that it is attempting to satisfy this promise. The delivery_plan.motive.planned_for_promise will be set 'true' which will cause it to try to satisfy this Item_Promise's quantity' and its 'owner' Delivery_Promise's due Dates.

Alternatively, Item_request.plan_to_satisfy can be performed to set the plan to attempt to satisfy the request rather than the promise. Or, the planned_for_promise field of the delivery_plan.motive can be set directly to select whether to plan for this promise or the Item_request.

Properties: command=True Export-Only Field

promise_as_planned - - *a Void field of model Item_Promise*

This command sets this Item_Promise's quantity according to the Quantity currently planned and what was requested.

The quantity is set to the Quantity_Range that includes all of Item_request.quantity and the planned Quantity. (The quantity.min is set to the lesser of the Item_request.quantity.min and the planned Quantity. The quantity.max is set to the greater of the Item_request.quantity.max and the planned Quantity)

Properties: command=True Export-Only Field

Models**Item_Promise Model**

plan_as_promised - - *a Void field of model Item_Promise*

This command creates or re-sizes the receiving_plan for this Item_RAP such that it satisfies this promise.

Properties: command=True Export-Only Field

receiving_plan - - *a Operation, Plan field of model Item_Promise*

The Operation that has been planned to receive this Item_Promise.

Properties: Export-Only Field

not_planned - - *a Logical field of model Item_Promise*

If "true", then no Operation has been planned to satisfy this Item_Promise. If "true", then there is a PROMISE_NOT_PLANNED Problem for this Item_Promise.

Properties: Export-Only Field

planned_late - - *a Time field of model Item_Promise*

If the plan for this Item_Promise is planned for later than promised, then this returns how much later; otherwise it returns "0". This is "0" if quantity is "0" or 'ownerdue' is infinite. If this is non-zero, then there is a PROMISE_PLANNED_LATE Problem for this Item_Promise.

Properties: Export-Only Field

planned_early - - *a Time field of model Item_Promise*

If the plan for this Item_Promise is planned for earlier than promised, then this returns how much earlier; otherwise it returns "0". This is "0" if quantity is "0" or 'ownerdue' is infinite. If this is non-zero, then there is a PROMISE_PLANNED_EARLY Problem for this Item_Promise.

Properties: Export-Only Field

planned_short - - *a Quantity field of model Item_Promise*

If the plan for this Item_Promise is planned to deliver less than promised, then this returns how much less; otherwise it returns "0". This is "0" if quantity is "0" or 'ownerdue' is infinite. If this is non-zero, then there is a PROMISE_PLANNED_SHORT Problem for this Item_Promise.

Properties: Export-Only Field

planned_excess - - *a Quantity field of model Item_Promise*

If the plan for this Item_Promise is planned to deliver more than promised, then this returns how much more; otherwise it returns "0". This is "0" if quantity is "0" or 'ownerdue' is infinite. If this is non-zero, then there is a PROMISE_PLANNED_EXCESS Problem for this Item_Promise.

Properties: Export-Only Field

Models	Item_Promise Model
--------	--------------------

last_change - - *a Date field of model Item_Promise*

The Date at which the most recent change was made to this Item_Promise. This is set by setting this or any other field in this model. Note that setting this directly will set it to the specified Date, which may not be the current Date.

This is used to support the 'changed_promises' functions of Site, which provide net-change exporting of Promises.

This is the same as calling owner.r_last_change.

Default: now

problems, problems (Date_Range) , problems (Problem_Category) , problems (Date_Range, Problem_Category) - - a List(Problem) field of model Item_Promise

The Problems associated with this Item_Promise. If passed a Date_Range, only the Problems whose dates overlap are returned. If passed a Problem_Category, only the Problems with that category are returned.

Properties: Export-Only Field

promised_quantity_problem - - a Problem field of model Item_Promise

Obsolete! Use Item_Promise.problems. The REQUEST_PROMISED_QUANTITY Problem, if any, that exists between this Item_Promise and the Item_request it is answering. See fields promised_short and promised_excess for more convenient ways to test for REQUEST_PROMISED_SHORT, and REQUEST_PROMISED_EXCESS Problems, respectively. This will be nonexistent if there is no such Problem or if owner.unanswered is "true".

Properties: obsolete=True Export-Only Field

promised_price_problem - - a Problem field of model Item_Promise

Obsolete! Use Item_Promise.problems. The REQUEST_PROMISED_OVERPRICED Problem, if any, that exists between this Item_Promise and the Item_request it is answering. See the field 'overpriced' for an alternate way to test for this Problem. This will be nonexistent if there is no such Problem or if owner.unanswered is "true".

Properties: obsolete=True Export-Only Field

Models	Item_Promise Model
--------	--------------------

planned_date_problem - - a Problem field of model Item_Promise

Obsolete! Use Item_Promise.problems. The PROMISE_PLANNED_DATE Problem, if any, that exists between this Item_Promise and the Operation_Plan currently planned to satisfy it (delivery_plan). See fields not_planned, planned_late, and planned_early for alternate ways to test for PROMISE_NOT_PLANNED, PROMISE_PLANNED_LATE, and PROMISE_PLANNED_EARLY Problems, respectively.

Note well: the lack of a Problem here only indicates that a plan to meet this Promise is being attempted. There may be feasibility Problems with the plans associated with fulfilling this Promise in addition to any Problem here.

Properties: obsolete=True Export-Only Field

planned_quantity_problem - - a Problem field of model Item_Promise

Obsolete! Use Item_Promise.problems. The PROMISE_PLANNED_QUANTITY Problem, if any, that exists between this Item_Promise and the Operation_Plan currently planned to satisfy it (delivery_plan). See fields not_planned, planned_short, and planned_excess for more convenient ways to test for PROMISE_NOT_PLANNED, PROMISE_PLANNED_SHORT, and PROMISE_PLANNED_EXCESS Problems, respectively.

Note well: the lack of a Problem here only indicates that a plan to meet this Promise is being attempted. There may be feasibility Problems with the plans associated with fulfilling this Promise in addition to any Problem here.

Properties: obsolete=True Export-Only Field

owner - - a Delivery_Promise field of model Item_Promise

Properties: Export-Only Field

Models	Item_Available_To_Promise Model
--------	---------------------------------

5.1.3.1.8.1.2.1 Item_Available_To_Promise Model

Item_Available_To_Promise -- *a submodel of model Item_Promise*

An Item_Available_To_Promise models the available quantity of an item on a specific date (the *Delivery_Available_To_Promise* date) to satisfy an Item_Request for the requested item. Therefore, it models a forecast Promise that is suitable for the *owner:Item_request*, that has been allocated for use by the Seller, but has not yet been consumed by an actual (customer) Request. Such a forecast Promise is then 'available' for being consumed to form a Promise for the 'owner' Item_Request. This Item_Available_To_Promise takes the form of a item date and a list of Item_Available_To_Promises which constitute the available promises for each Item_Request in *owner:Item_request*:

Note that there may be numerous Products that match the *owner:Item_request's* Item_Request. Given that, there may be numerous different forecast Promises that are available to Promise the *owner:Item_request's* Item_Request. They may vary in Quantity, Item Date, Configuration, and price. See the Product model for more detail on how an Item_Request is matched with a Product.

Furthermore, if the available allocation to the Seller for a particular Product is inadequate, the Seller can look at its organization's allocations for that Product.

Note that the "availability" recorded by these models can be fleeting. If more than one User has access to the same Seller allocations (particularly for organization's), then allocations can be consumed between the time that the Item_Available_To_Promise is computed and the decision is made to consume it. Or plan revisions may be made in that same time, causing allocations to disappear. How stable ATP computations are depends upon the environment and usage. By allocating to individual Users and by insulating the order quoting environment from plan changes, the ATP computations can be made very stable.

The Item_Available_To_Promise model has these submodels :
Product_Available_To_Promise.

The Item_Available_To_Promise model has fields that references these models :
Product_Available_To_Promise, Item_Promise.

These models have a field that is a Item_Available_To_Promise model :
Item_Promise, Product_Available_To_Promise.

The key field for this model is dates

Models	Item_Available_To_Promise Model
--------	---------------------------------

dates -- *a Date_Range field of model Item_Available_To_Promise*
The *due Date_Range* that could be Promised with this Item_Available_To_Promise.
Default: None -- this is a key field
Properties: Export-Only Field

available -- *a Quantity field of model Item_Available_To_Promise*
The amount available to promise for *owner:Item_request*. This quantity is computed according to the following rule: *min(owner:Item_request:quantity:min, product_atp_for_each(#quantity) sum)*
Properties: Export-Only Field

product_atp -- *a list of Product_Available_To_Promise submodels of model Item_Available_To_Promise*
The list of possibilities for promising 'owner' on 'dates' for this Seller.

offer_promise -- *a Void field of model Item_Available_To_Promise*
Consume this Item_Available_To_Promise in order to form a Promise for 'owner'. If this Item_Available_To_Promise has inadequate Quantity for its 'owner', then this will result in a new Item_Request with this Item_Available_To_Promise's available quantity. The appropriate *owner:Item_request's* item_request will be reduced in Quantity by like amount.

If the 'dates' of this satisfy the *owner:Item_request*, then the new Item_Request(s) will have the same Item_Request 'owner', and thus will be delivered together. However, if 'dates' does not satisfy it, then a new Item_Request will be created as well.

Properties: command=True Export-Only Field

owner -- *a Item_Promise field of model Item_Available_To_Promise*

Properties: Export-Only Field

5.13.1.8.12.1.1 Product_Available_To_Promise Model

Product_Available_To_Promise -- *a submodel of model Item_Available_To_Promise*

A Product_Available_To_Promise models the quantity available for each equivalent product, since more than one product may be able to satisfy the request for the item. Therefore, a forecast Promise that is suitable for the 'owner' Item_Request, that has been allocated for use by the Seller, but has not yet been consumed by an actual (customer) Request. Such a forecast Promise is then "available" for being consumed to form a Promise for the 'owner' Item_Request.

Note that there may be numerous Products that match the 'owner' Item_Request. Given that, there may be numerous different forecast Promises that are available to Promise the 'owner' Item_Request. They may vary in Quantity, delivery Date, Configuration, and price. See the Product model for more detail on how an Item_Request is matched with a Product.

Furthermore, if the available allocation to the Seller for a particular Product is inadequate, the Seller can look at its organization's allocations for that Product.

Note that the "availability" recorded by these models can be fleeting. If more than one User has access to the same Seller allocations (particularly for organization's), then allocations can be consumed between the time that the Product_Available_To_Promise is computed and the decision is made to consume it. Or plan revisions may be made in that same time, causing allocations to disappear. How stable ATP computations are depends upon the environment and usage. By allocating to individual Users and by insulating the order quoting environment from plan changes, the ATP computations can be made very stable.

The Product_Available_To_Promise model has fields that references these models: Product, Forecast, Forecast_Entry, Item_Available_To_Promise.

These models have a field that is a Product_Available_To_Promise model: Item_Available_To_Promise.

The key field for this model is product

product -- *a Product field of model Product_Available_To_Promise*
The Product for which there is this Product_Available_To_Promise.

This is not really settable (doc to be fixed).
Default: None -- this is a key field

Properties: Export-Only Field

forecast -- *a Forecast field of model Product_Available_To_Promise*
The Forecast for product from which this Product_Available_To_Promise was computed.
Properties: Export-Only Field

forecast_entry -- *a Forecast_Entry field of model Product_Available_To_Promise*
The latest Forecast_Entry for product from which this Product_Available_To_Promise was computed.
Properties: Export-Only Field

dates -- *a Date_Range field of model Product_Available_To_Promise*
The due Date_Range that could be Promised with this Product_Available_To_Promise.
Properties: Export-Only Field

price -- *a Money field of model Product_Available_To_Promise*
The price that could be quoted for this Product_Available_To_Promise.
Properties: Export-Only Field

quantity -- *a Quantity field of model Product_Available_To_Promise*
The Quantity that is Product_Available_To_Promise to the 'owner' Item_Request from the 'forecast_entry'. This Quantity is converted to the 'unit' of the 'item' of the 'owner' Item_Request.
Properties: Export-Only Field

offered_quantity -- *a Quantity field of model Product_Available_To_Promise*
The Quantity that will be promised to the 'owner' Item_Request by offered_promise.
This Quantity is converted to the 'unit' of the 'item' of the 'owner' Item_Request.
Default: value->available()

offer_promise_offered_promise (Quantity) -- *a Void field of model Product_Available_To_Promise*
Consume this Product_Available_To_Promise in order to form a Promise for 'owner'. If this Item_Available_To_Promise has inadequate Quantity, then this will result in a new Item_Request with this Item_Available_To_Promise's Quantity. The 'owner' Item_Request will be reduced in Quantity by like amount.

If the 'dates' of this satisfy the 'owner's Delivery_Request, then the new Item_Request will have the same Delivery_Request 'owner', and thus will be delivered together. However, if 'dates' does not satisfy it, then a new Delivery_Request will be created as well.

The 'owner.item.request.quantity.max' will be reduced by the Quantity consumed to satisfy the 'owner'. Of course, no more than 'owner.item.request.quantity.max' will be consumed, so 'owner.item.request.quantity.max' will never be less than 0.

Properties: command=True Export-Only Field

owner -- a Item_Available_To_Promise field of model
Product_Available_To_Promise

5.1.3.1.9 Acceptance Model

Acceptance -- a submodel of model Site_Plan

An Acceptance models the 'owner' requester's response to a Supplier's Promise. Note that a Acceptance model exists for every Request/Promise model. However, the Acceptance may not be 'accepted' yet.

The Acceptance model has these submodels :
Delivery_Acceptance.

The Acceptance model has fields that references these models :
Promise, Delivery_Acceptance, Site_Plan.

These models have a field that is a Acceptance model :
LINK, Promise, SUPPLIER, Delivery_Acceptance,
REQUEST_NOT_PLANNED, REQUEST_PLANNED_LATE,
REQUEST_PLANNED_EARLY, REQUEST_PLANNED_SHORT,
REQUEST_PLANNED_EXCESS, PROMISE_NOT_PLANNED,
PROMISE_PLANNED_LATE, PROMISE_PLANNED_EARLY,
PROMISE_PLANNED_SHORT, PROMISE_PLANNED_EXCESS,
ACCEPTANCE_NOT_PLANNED, ACCEPTANCE_PLANNED_LATE,
ACCEPTANCE_PLANNED_EARLY, ACCEPTANCE_PLANNED_SHORT,
ACCEPTANCE_PLANNED_EXCESS, REQUEST_PROMISED_SHORT,
REQUEST_PROMISED_EARLY, REQUEST_PROMISED_LATE,
REQUEST_PROMISED_EXCESS, ITEM_PROMISE_OVERPRICED,
DELIVERY_PROMISE_OVERPRICED, PROMISE_NOT_OFFERED,
PROMISE_NOT_CONFIRMED, PROMISE_NOT_ACCEPTED,

ACCEPTANCE_INCONSISTENT, REQUEST_QUEUED,
DELIVERY_REQUEST_NOT_COORDINATED,
DELIVERY_PROMISE_NOT_COORDINATED,
DELIVERY_ACCEPTANCE_NOT_COORDINATED,
SUPPLY_PLANNED_LATE, SUPPLY_PLANNED_EARLY,
SUPPLY_PLANNED_SHORT, SUPPLY_PLANNED_EXCESS,
SUPPLY_PROMISED_LATE, SUPPLY_PROMISED_EARLY,
SUPPLY_PROMISED_SHORT, SUPPLY_PROMISED_EXCESS.

The key field for this model is promise
This model may be extended with user-defined fields.

promise -- a Promise field of model Acceptance
The Promise to which this Acceptance responds.

Default: None -- this is a key field
Properties: Export-Only Field

delivery_acceptances -- a list of Delivery_Acceptance submodels of model Acceptance
The individual acceptances for delivery_promises.

accepted -- a Date field of model Acceptance
The Date that the requester accepted the promise
Default: oo_past

accept_by -- a Date field of model Acceptance
The Date by which the 'customer' intends to accept or reject any Promise that is made.
This is the Date on which both parties would become committed. And it is the start of the delivery_lead_time (which can affect pricing).

Note that this is just the requested 'accept_by' Date -- the actual deadline for accepting a Promise is in the Promise 'accept_by'. The supplier considers this requested Date when making the Promise, but may choose a different Date based upon the delivery lead times of its Products. Such differences are part of the negotiation of Request and Promise.

Default: promise.request.promise_by + "1 day"

plan_to_satisfy -- a Void field of model Acceptance
This command creates plans to satisfy this Acceptance. It does this by simply calling plan_to_satisfy on each of its 'delivery_acceptances'. Thus, it is equivalent to delivery_acceptances.for_each{#plan_to_satisfy}.

Alternatively, 'request.plan_to_satisfy' can be performed to call 'plan_to_satisfy' on each of the 'delivery_requests'.

Properties: command=True Export-Only Field

plan_as_accepted -- *a Void field of model Acceptance*

This command sets the plan for each of this Acceptance's 'item_acceptances'.

Properties: command=True Export-Only Field

accept_as_promised, accept_as_promised (Logical) -- *a Void field of model Acceptance*

This command

sets this Acceptance to accept what has been promised. It does this by simply calling 'accept_as_promised' on each of its 'delivery_acceptance'. Thus, it is equivalent to 'delivery_acceptances.for_each(&accept_as_promised)'.

Properties: command=True Export-Only Field

last_change -- *a Date field of model Acceptance*

The Date at which the most recent change was made to this Acceptance. This is set by setting this or any other field in this model or the 'delivery_acceptances'. Note that setting this directly will set it to the specified Date, which may not be the current Date.

This is used to support the changed_acceptances' functions of Site, which provide net-change Exporting of Acceptances.

Default: now

problems, problems (Date_Range) , problems (Problem_Category) , problems (Date_Range, Problem_Category) -- *a List(Problem) field of model Acceptance*
The Problems associated with this Acceptance, including any problems that are associated with the underlying Delivery Acceptance and Item Acceptance models. If passed a Date_Range, only the Problems whose 'dates' overlap are returned. If passed a Problem_Category, only the Problems with that category are returned.

Properties: Export-Only Field

owner -- *a Site_Plan field of model Acceptance*

Properties: Export-Only Field

5.1.3.1.9.1 Delivery_Acceptance Model

Delivery_Acceptance -- *a submodel of model Acceptance*

Delivery_Acceptance models the supplier's response to a customer's Delivery_Request for a set of Items to be delivered together. Several Delivery_Acceptances may make up the 'owner' Acceptance.

The model has selectors:
fulfillment_policy.

The Delivery_Acceptance model has these submodels:
Item_Acceptance.

The Delivery_Acceptance model has fields that reference these models:
Delivery_Promise, Item_Acceptance, Problem.

These models have a field that is a Delivery_Acceptance model:
Acceptance, Delivery_Promise, Item_Acceptance,
REQUEST_NOT_PLANNED, REQUEST_PLANNED_LATE,
REQUEST_PLANNED_EARLY, REQUEST_PLANNED_SHORT,
REQUEST_PLANNED_EXCESS, PROMISE_NOT_PLANNED,
PROMISE_PLANNED_LATE, PROMISE_PLANNED_EARLY,
PROMISE_PLANNED_SHORT, PROMISE_PLANNED_EXCESS,
ACCEPTANCE_NOT_PLANNED, ACCEPTANCE_PLANNED_LATE,
ACCEPTANCE_PLANNED_EARLY, ACCEPTANCE_PLANNED_SHORT,
ACCEPTANCE_PLANNED_EXCESS, REQUEST_PROMISED_LATE,
REQUEST_PROMISED_EARLY, REQUEST_PROMISED_SHORT,
REQUEST_PROMISED_EXCESS, ITEM_PROMISE_OVERPRICED,
DELIVERY_PROMISE_OVERPRICED, PROMISE_NOT_CONFIRMED,
DELIVERY_REQUEST_NOT_COORDINATED,
DELIVERY_PROMISE_NOT_COORDINATED,
DELIVERY_ACCEPTANCE_NOT_COORDINATED,
SUPPLY_PLANNED_LATE, SUPPLY_PLANNED_EARLY,
SUPPLY_PLANNED_SHORT, SUPPLY_PLANNED_EXCESS,
SUPPLY_PROMISED_LATE, SUPPLY_PROMISED_EARLY,
SUPPLY_PROMISED_SHORT, SUPPLY_PROMISED_EXCESS.

The key field for this model is delivery_promise
This model may be extended with user-defined fields.

delivery_promise - - - *a Delivery_Promise field of model Delivery_Acceptance*
The **Delivery_Promise** to which this **Delivery_Acceptance** responds.

Default: None - - this is a key field

Properties: Export-Only Field

item_acceptances - - - *a list of Item_Acceptance submodels of model Delivery_Acceptance*
Acceptances for the individual order-line-items.

due - - - *a Date_Range field of model Delivery_Acceptance*

The **Date_Range** within which the promise is accepted by the customer.

Default: **delivery_promise.due**

fulfillment_policy - - - *an extension selector of model Delivery_Acceptance*

This policy defines whether a delivery can be shorted or delayed.

Typically this is the same as **delivery_promise.fulfillment_policy**. However, if the customer is not willing to do the promised **fulfillment_policy**, the acceptance one may be different.

Default: **delivery_promise.fulfillment_policy**

Extensions:

ON_TIME, FULL_QUANTITIES_OF_ALL_ITEMS, UNRESTRICTED.

plan_to_satisfy - - - *a Void field of model Delivery_Acceptance*

This command sets the plan for each of this **Delivery_Acceptance's** **item_acceptances**.

Properties: command=True Export-Only Field

plan_as_accepted - - - *a Void field of model Delivery_Acceptance*

This command sets the plan for each of this **Delivery_Acceptance's**

item_acceptances.

Properties: command=True Export-Only Field

accept_as_promised - - - *a Void field of model Delivery_Acceptance*

This command sets this **Delivery_Acceptance** and its **Item_Acceptances** according to what is currently promised

Properties: command=True Export-Only Field

not_planned - - - *a Logical field of model Delivery_Acceptance*

If "true", then no plan has been formed to meet at least parts of this

Delivery_Acceptance. This is "true" if "due" is finite (unanswered is "false") and at least one of the **item_acceptances** is unplanned. If "true", then there is a

ACCEPTANCE_NOT_PLANNED Problem for those unplanned **Item_Promises**.

Properties: Export-Only Field

planned_late - - - *a Time field of model Delivery_Acceptance*

Returns how late this **Delivery_Acceptance** is currently planned to be fully delivered.

It is the latest of each of the **Item_acceptances**. Note that

delivery_acceptance.planned_late is equivalent to

max(delivery_acceptance.item_acceptances.for_each(#planned_late)). If non-zero,

then there is a **ACCEPTANCE_PLANNED_LATE** Problem for at least one of the **item_acceptances**.

Unlike the individual **item_acceptances**, a **Delivery_Acceptance** can have both **planned_late** and **planned_early** non-zero.

Properties: Export-Only Field

planned_early - - - *a Logical field of model Delivery_Acceptance*

Returns how early this **Delivery_Acceptance** is currently planned to be delivered, at

least in part. It is the earliest of each of the **Item_acceptances**. Note that

delivery_acceptance.planned_early is equivalent to

min(delivery_acceptance.item_acceptances.for_each(#planned_early)). If non-zero,

then there is a **ACCEPTANCE_PLANNED_EARLY** Problem for at least one of the **item_acceptances**.

Unlike the individual **item_acceptances**, a **Delivery_Acceptance** can have both

planned_late and **planned_early** non-zero.

Properties: Export-Only Field

last_change - - - *a Date field of model Delivery_Acceptance*

The **Date** at which the most recent change was made to this **Delivery_Acceptance**.

This is set by setting this or any other field in this model or the **Item_acceptances**.

Note that setting this directly will set it to the specified **Date**, which may not be the current **Date**.

This is used to support the **changed_acceptances** functions of **Site**, which provide net-change Exporting of **Acceptances**.

Default: now

problems, problems (Date_Range) , problems (Problem_Category) , problems (Date_Range, Problem_Category) -- a List(Problem) field of model

Delivery_Acceptance

The Problems associated with this Delivery_Acceptance, including any problems that are associated with the underlying Item_Acceptance model. If passed a Date_Range, only the Problems whose dates overlap are returned. If passed a Problem_Category, only the Problems with that category are returned.

Properties: Export-Only Field

plan_problems -- a List(Problem) field of model Delivery_Acceptance
Obsolete! Use Delivery_Acceptance.problems. A List of the ACCEPTANCE_PLANNED Problems of the Item_Acceptances, if any. See fields unplanned, planned_late, and planned_early for alternate ways to test for ACCEPTANCE_NOT_PLANNED, ACCEPTANCE_PLANNED_LATE, and ACCEPTANCE_PLANNED_EARLY Problems, respectively.

Note well: the lack of a Problem here only indicates that a plan to meet this Acceptance is being attempted. There may be feasibility Problems with the plans associated with fulfilling this Acceptance in addition to any Problem here.

Properties: obsolete=True Export-Only Field

delivery_not_coordinated_problem -- a Problem field of model Delivery_Acceptance

Obsolete! Use Delivery_Acceptance.problems. The

DELIVERY_ACCEPTANCE_NOT_COORDINATED Problem, if any, that exists

Properties: obsolete=True Export-Only Field

owner -- a Acceptance field of model Delivery_Acceptance

Properties: Export-Only Field

5.1.3.19.1.1 Item_Acceptance Model

Item_Acceptance -- a submodel of model Delivery_Acceptance

An Item_Acceptance is an agreement to supply/consume a Quantity of a particular Item. The Date_Range within which this Quantity should be supplied is given by the 'owner' Delivery_Acceptance, which can also coordinate multiple Item_Acceptances together.

The Item_Acceptance model has fields that references these models :
Item_Promise, Configuration, Operation_Plan, Problem, Delivery_Acceptance.

These models have a field that is a Item_Acceptance model :
Delivery_Acceptance, Item_Promise, REQUEST_NOT_PLANNED, REQUEST_PLANNED_LATE, REQUEST_PLANNED_EARLY, REQUEST_PLANNED_SHORT, REQUEST_PLANNED_EXCESS, PROMISE_NOT_PLANNED, PROMISE_PLANNED_LATE, PROMISE_PLANNED_EARLY, PROMISE_PLANNED_SHORT, PROMISE_PLANNED_EXCESS, ACCEPTANCE_NOT_PLANNED, ACCEPTANCE_PLANNED_LATE, ACCEPTANCE_PLANNED_EARLY, ACCEPTANCE_PLANNED_SHORT, ACCEPTANCE_PLANNED_EXCESS, REQUEST_PROMISED_SHORT, REQUEST_PROMISED_EXCESS, ITEM_PROMISE_OVERPRICED, SUPPLY_PLANNED_LATE, SUPPLY_PLANNED_EARLY, SUPPLY_PLANNED_SHORT, SUPPLY_PLANNED_EXCESS, SUPPLY_PROMISED_LATE, SUPPLY_PROMISED_EARLY, SUPPLY_PROMISED_SHORT, SUPPLY_PROMISED_EXCESS.

The key field for this model is item_promise

item_promise -- a Item_Promise field of model Item_Acceptance
The Item_Request that this Item_Promise is trying to satisfy.

Default: None -- this is a key field

Properties: Export-Only Field

configuration -- a Configuration field of model Item_Acceptance
The Item being requested and the specific characteristics desired. The Item may be a STANDARD Item (no characteristics to specify) a Request-specific configuration of a OPTIONED Item, or a Request-specific CUSTOM Item (among others).

This is almost always the same as `Item_Request.Configuration` (unless the requested configuration is not available or feasible). This is not scellable itself, but you can set the fields of this Configuration.

Properties: `command=True` Export-Only Field

Item `-- a Item field of model Item_Acceptance`
This is simply a shortcut to `Configuration.Item`.

If the Item is [unspecified] (the default), then this is considered to be a promise for nothing.

Default: [unspecified]

quantity `-- a Quantity_Range field of model Item_Acceptance`

The range of Quantity of the Configuration that is being accepted as part of the owner's Delivery_Acceptance. This is typically `Item_Request.Quantity`. However, if that Quantity was not feasible, then this Quantity may be less.

These Quantity's are converted to the unit of the Item being requested.

Default: `Item.Promise.Quantity`

delivery_plan `-- a Operation_Plan field of model Item_Acceptance`

The Operation that has been planned to deliver this Item_Acceptance. It is the same as `Item_Request.Delivery_Plan`.

Properties: Export-Only Field

plan_to_satisfy `-- a Void field of model Item_Acceptance`

This command sets the plan for this Item_Acceptance such that it is attempting to satisfy this acceptance. The `delivery_plan.motive.planned_for_acceptance` will be set "true" which will cause it to try to satisfy this Item_Acceptance's quantity and its owner's Delivery_Acceptance's due Dates.

Properties: `command=True` Export-Only Field

accept_as_promised `-- a Void field of model Item_Acceptance`

This command sets this Item_Acceptance's quantity according to the Quantity currently planned and what was requested.

The quantity is set to the Quantity_Range that includes all of `Item_Request.Quantity` and the planned Quantity. (The quantity_min is set to the lesser of the `Item_Request.Quantity_Min` and the planned Quantity. The quantity_max is set to the greater of the `Item_Request.Quantity_Max` and the planned Quantity.)
Properties: `command=True` Export-Only Field

plan_as_accepted `-- a Void field of model Item_Acceptance`
This command creates or re-sizes the receiving_plan for this Item_RAP such that it satisfies this acceptance.

Properties: `command=True` Export-Only Field

not_planned `-- a Logical field of model Item_Acceptance`

If "true", then no Operation has been planned to satisfy this Item_Acceptance. If "true", then there is a `ACCEPTANCE_NOT_PLANNED` Problem for this Item_Acceptance.

Properties: Export-Only Field

planned_late `-- a Time field of model Item_Acceptance`

If the plan for this Item_Acceptance is planned for later than accepted, then this returns how much later; otherwise it returns "0". This is "0" if quantity is "0" or owner's due is infinite. If this is non-zero, then there is a `ACCEPTANCE_PLANNED_LATE` Problem for this Item_Acceptance.
Properties: Export-Only Field

planned_early `-- a Time field of model Item_Acceptance`

If the plan for this Item_Acceptance is planned for earlier than accepted, then this returns how much earlier; otherwise it returns "0". This is "0" if quantity is "0" or owner's due is infinite. If this is non-zero, then there is a `ACCEPTANCE_PLANNED_EARLY` Problem for this Item_Acceptance.
Properties: Export-Only Field

planned_short `-- a Quantity field of model Item_Acceptance`

If the plan for this Item_Acceptance is planned to deliver less than accepted, then this returns how much less; otherwise it returns "0". This is "0" if quantity is "0" or owner's due is infinite. If this is non-zero, then there is a `ACCEPTANCE_PLANNED_SHORT` Problem for this Item_Acceptance.
Properties: Export-Only Field

planned_excess `-- a Quantity field of model Item_Acceptance`

If the plan for this Item_Acceptance is planned to deliver more than accepted, then this returns how much more; otherwise it returns "0". This is "0" if quantity is "0" or owner's due is infinite. If this is non-zero, then there is a `ACCEPTANCE_PLANNED_EXCESS` Problem for this Item_Acceptance.
Properties: Export-Only Field

last_change - - *a Date field of model Item_Acceptance*
The Date at which the most recent change was made to this Item_Acceptance. This is set by setting this or any other field in this model. Note that setting this directly will set it to the specified Date, which may not be the current Date.

This is used to support the changed_acceptances functions of Site, which provide net-change exporting of Acceptances.

This is the same as calling owner.r_last_change.

Default: now

problems, problems (Date_Range) , problems (Problem_Category) , problems (Date_Range, Problem_Category) - - a List(Problem) field of model Item_Acceptance

The Problems associated with this Item_Acceptance if passed a Date_Range, only the Problems whose dates overlap are returned. If passed a Problem_Category, only the Problems with that category are returned.

Properties: Export-Only Field

planned_date_problem - - a Problem field of model Item_Acceptance
Obsolete! Use Item_Acceptance.problems. The ACCEPTANCE_PLANNED_DATE Problem, if any, that exists between this Item_Acceptance and the Operation_Plan currently planned to satisfy it (delivery_plan). See fields 'not_planned', 'planned_late', and 'planned_early' for alternate ways to test for ACCEPTANCE_NOT_PLANNED, ACCEPTANCE_PLANNED_LATE, and ACCEPTANCE_PLANNED_EARLY Problems, respectively.

Note well: the lack of a Problem here only indicates that a plan to meet this Acceptance is being attempted. There may be feasibility Problems with the plans associated with fulfilling this Acceptance in addition to any Problem here.

Properties: obsolete=True Export-Only Field

planned_quantity_problem - - a Problem field of model Item_Acceptance
Obsolete! Use Item_Acceptance.problems. The ACCEPTANCE_PLANNED_QUANTITY Problem, if any, that exists between this Item_Acceptance and the Operation_Plan currently planned to satisfy it (delivery_plan). See fields 'not_planned', 'planned_short', and 'planned_excess' for more convenient ways to test for ACCEPTANCE_NOT_PLANNED, ACCEPTANCE_PLANNED_SHORT, and ACCEPTANCE_PLANNED_EXCESS Problems, respectively.

Note well: the lack of a Problem here only indicates that a plan to meet this Acceptance is being attempted. There may be feasibility Problems with the plans associated with fulfilling this Acceptance in addition to any Problem here.

Properties: obsolete=True Export-Only Field

owner - - a Delivery_Acceptance field of model Item_Acceptance

Properties: Export-Only Field

5.1.3.2 Seller_Plan Model

Seller_Plan -- a submodel of model Plan

The master plan for a Seller. It includes the Seller's Forecast of how much of each of its Products could be sold, the sales the Seller has committed to making, the allocations made from the supplying Sites to the Seller, and the customer Promises the Seller has made based on those allocations.

The Seller_Plan model has these submodels :
Forecast.

The Seller_Plan model has fields that references these models :
Seller, Seller_Plan, Forecast.

These models have a field that is a Seller_Plan model :
Seller, Plan, Forecast, Seller_Plan, Request, Delivery_Request.

The key field for this model is seller

This model may be extended with user-defined fields.

seller -- a Seller field of model Seller_Plan
The Seller that this Seller_Plan is planning.

This field is not really settable (to be fixed).
Default: None -- this is a key field

organization -- a Seller_Plan field of model Seller_Plan
The Seller_Plan for this seller's organization.
Properties: Export-Only Field

members -- a List(Seller_Plan) field of model Seller_Plan
This List contains the Seller_Plan for each of this seller's immediate 'members'. Use 'seller:recurse(#.members)' to get the entire tree below 'seller'.
Properties: Export-Only Field

forecasts -- a list of Forecast submodels of model Seller_Plan
The forecasts and allocations (the master plans) for each Product and Product_Group of this Seller and its 'organizations' (e.g., 'all_products' + 'all_product_groups'). These Forecasts form a hierarchy that parallels the Seller's Product_Group hierarchy. The set of INDIVIDUAL_Forecasts constitutes the master plan for the Seller. The GROUP_Forecasts are essentially a tool for aggregating and disaggregating the INDIVIDUAL_Forecasts.

active_forecasts -- a List(Forecast) field of model Seller_Plan
The 'active' forecasts for this Seller Plan (see Forecast.active). The returned list will include all GROUP forecasts.
Properties: Export-Only Field

forecast (Symbol) -- a Forecast field of model Seller_Plan
The Forecast of this Seller_Plan for the Product or Product_Group with the specified name. That Product or Product_Group may be defined by this Seller, or one of its organizations.

Note that 'forecast(name)' is equivalent to 'forecasts.find(name)', but is simpler, clearer, and more efficient. Also note that if a Product and a Product_Group have the same name, this will return the Forecast for the Product_Group. Use 'product_forecast(name)' to get the Product's Forecast.
Properties: Export-Only Field

top_forecasts -- a List(Forecast) field of model Seller_Plan
The Forecasts for the Seller's 'top_products' and 'top_product_groups', those that are the top of a Product_Group hierarchy. These are the Forecasts that do not appear within any other GROUP Forecast. The [unspecified] Forecast is not included.
Properties: Export-Only Field

active_top_forecasts -- a List(Forecast) field of model Seller_Plan
The active Forecasts for the Seller's 'top_products' and 'top_product_groups'.
Properties: Export-Only Field

product_root_forecasts -- a List(Forecast) field of model Seller_Plan
The INDIVIDUAL_Product_Root Forecasts (Forecasts for which this Seller defines the root). The [unspecified] Forecast is not included.
Properties: Export-Only Field

active_product_root_forecasts -- a List(Forecast) field of model Seller_Plan
The active INDIVIDUAL_Product_Root Forecasts.

<i>Models</i>	<i>Seller_Plan Model</i>
---------------	--------------------------

Properties: Export-Only Field

product_forecasts - - *a List(Forecast) field of model Seller_Plan*
The INDIVIDUAL Product Forecasts. The (unspecified) Forecast is not included.
Properties: Export-Only Field

active_product_forecasts - - *a List(Forecast) field of model Seller_Plan*
The active INDIVIDUAL Product Forecasts.
Properties: Export-Only Field

product_forecast (Symbol) - - *a Forecast field of model Seller_Plan*
The INDIVIDUAL Product Forecast of this Seller_Plan for the Product with the specified name. That Product may be defined by this Seller, or one of its organizations.

Note that 'product_forecast(name)' is equivalent to 'product_forecasts.find(name)', but is simpler, clearer, and more efficient.
Properties: Export-Only Field

forecast_horizon - - *a List(Date_Range) field of model Seller_Plan*
The "forecasting buckets". The List of Date_Ranges that define the forecast horizon, that define the Date_Range of each Forecast_Entry. This is defined by the 'forecast_horizon' extension of the Seller.

Note that a horizon will always consist of a consecutive series of adjacent, but non-overlapping, Date_Ranges that cover the full planning horizon.
Properties: Export-Only Field

atp_horizon - - *a List(Date_Range) field of model Seller_Plan*
The "allocation buckets" or "ATP buckets". The List of Date_Ranges that define the atp horizon, that define the Date_Range of each ATP_Entry within a Forecast for this Seller_Plan. This is defined by the 'atp_horizon' extension of this Seller_Plan's Seller.

Note that a horizon will always consist of a consecutive series of adjacent, but non-overlapping, Date_Ranges that cover the full planning horizon.
Properties: Export-Only Field

actual_requests - - *a List(Request) field of model Seller_Plan*
The customer Requests that have been placed through this Seller.
Properties: Export-Only Field

<i>Models</i>	<i>Seller_Plan Model</i>
---------------	--------------------------

actual_promises - - *a List(Promise) field of model Seller_Plan*
The customer Promises that have been made through this Seller.
Properties: Export-Only Field

problems, problems (Date_Range) , problems (Problem_Category) , problems (Date_Range, Problem_Category) - - *a List(Problem) field of model Seller_Plan*
The SELLER Problems associated with the Forecast_Entries of this Seller_Plan.
Properties: Export-Only Field

accept_as_allocated - - *a Void field of model Seller_Plan*
Accept the allocated values in all of the Forecast_Entry's of all the 'Forecasts' in this Seller_Plan and its members'. This does 'accept_as_allocated' for each of the 'Forecasts', which in turn does 'accept_as_allocated' for each of its 'entries', which copies the 'allocated' and 'retain_from_allocated' fields to the 'accepted' and 'retain_from_accepted' fields in that Forecast_Entry and in the corresponding Forecast_Entry(s) in every Forecast in the specific and member trees below it. It also sets 'date_accepted' in all of those Forecast_Entry's to 'now'.

Note that the full hierarchy is done because it is in general a bad idea to mix one set of allocations with another -- they may be completely inconsistent. However, you can always set any or all of the fields directly.
Properties: command=True Export-Only Field

owner - - *a Plan field of model Seller_Plan*
Properties: Export-Only Field

5.1.3.2.1 Forecast Model

Forecast -- a submodel of model Seller_Plan

In each Seller_Plan there is one Forecast for each Product and for each Product_Group defined by that Seller or its organizations. Each Forecast is made up of a series of Forecast_Entry's which each contain a raw 'forecasted' value, a 'committed' forecast value, an 'allocated' value, a 'consumed' value, and thus ATP values. There is one Forecast_Entry for each time period defined by the Seller's 'Forecast_horizon'.

Forecasts are effectively organized into two hierarchies: the Seller hierarchy as defined by the Seller 'organization' field and the Product_Group hierarchy defined by the 'product_groups' of the Seller. Since the Product_Group hierarchy is defined for each Seller, the Seller hierarchy is in a sense the "outer" hierarchy.

The 'level' extension of Forecast indicates whether it is an INDIVIDUAL Forecast (for a Product) or a GROUP Forecast (for a Product_Group). Note that the set of INDIVIDUAL Forecasts constitute the master plan for the Seller. The GROUP Forecasts are essentially a tool for aggregating and disaggregating the INDIVIDUAL Forecasts. Such aggregation is the essence of intelligent forecast management, since forecasts are much more accurate in aggregate.

However, the GROUP Forecasts are not limited to aggregation. Setting a GROUP Forecast_Entry's 'forecasted' or 'committed' fields to a new value will set the same field of each of its sub_forecasts' such that the sum of those sub_forecasts' equals the value set there. The property that the GROUP Forecast's values are the sums of its sub_forecasts' is always maintained. In that sense, a GROUP Forecast is just a tool for seeing the aggregated total and for adjusting a group of Forecast values evenly.

Setting a Forecast_Entry field immediately propagates the change up to any GROUP Forecasts that contain it, and immediately disaggregates the change down to any sub_forecasts. How the change is disaggregated depends upon the value of the use_split' field. If "false", then the change is split out in the same proportions that the sub_forecasts' currently have (thereby not changing the percentage splits). If "true", then the change is split according to the sub_splits in the Product definitions, regardless of the current distribution.

The Product_Group hierarchies are a tool for a single Seller to manipulate the Forecasts that he or she owns. As such, the 'forecasted' and 'committed' values in the Product_Group hierarchy are tied directly to one another -- changes are immediately propagated throughout the hierarchies.

In contrast, the Seller hierarchy is about coordination of multiple independent Sellers, who may each have responsibility, and therefore control, over their own forecasts, commitments, and allocations. In this case, the 'forecasted' and 'committed' fields are not tied directly to one another. Rather, an organization has 'member_forecasted' and 'member_committed' fields which are the sums of its 'member' Seller's values. These sums are independent of (though related and compared to) its own values. As such, disaggregation is not performed for 'forecasted' and 'committed' values.

Disaggregation through the Seller hierarchy is performed for the 'allocated' field. It is done according to the 'allocation_policy' in the Allocation_Policy of this Seller for this Product.

The model has selectors:
level.

The Forecast model has these submodels:
Forecast_Entry.

The Forecast model has fields that references these models:
Forecast, Request, Forecast_Entry, Seller_Plan.

These models have a field that is a Forecast model:
Plan, Forecast, Seller_Plan, Request, Forecast_Entry, ATP_Entry, Item_Promise, Product_Available_To_Promise.

The key field for this model is name
This model may be extended with user-defined fields.

level -- an extension selector of model Forecast
If "GROUP", then this Forecast is for a Product_Group, an aggregation of several Products' Forecasts. The Product_Group defines how this Forecast is propagated down to sub_groups' or 'products'.

If "INDIVIDUAL", then this Forecast is for an individual Product. The Product defines how this Forecast is converted into Requests upon this Slice.

Default: INDIVIDUAL
Properties: Export-Only Field
Extensions:
INDIVIDUAL, GROUP.

name - - *a Symbol/field of model Forecast*
If 'level' is 'GROUP', then this field is `product_group.name`, otherwise it is `product.name`. This field is not settable.
Default: None - - this is a key field
Properties: Export-Only Field

remark - - *a String/field of model Forecast*
Any remarks about this Forecast.
Default: none

active - - *a Logical/field of model Forecast*
A Forecast is 'active' if any of it's fields has been set to other than the default value.
The conversion to "active" can occur either through directly setting a specific forecast or by setting a higher level GROUP Forecast. GROUP Forecasts will always be inactive since setting a field in a GROUP Forecast simply sets the fields of the members of the group.

The default for each of the Forecast model fields which return `List(Forecast)` is to return the complete set of Forecasts (both 'inactive' and 'active'). Each such field will also have an 'active' counterpart. For example, `Forecast.member_forecasts()` returns all Forecasts of the member Sellers, but `Forecast.active_member_forecasts()` returns only those Forecasts actively in use by the member Sellers.

The 'active' vs 'inactive' distinction is useful in situations where the number of possible Seller/Product Forecast combinations is much larger than the number of Seller/Product combinations actively being forecasted.

The 'active' field is not settable.
Properties: Export-Only Field

root - - *a Forecast/field of model Forecast*
The root Forecast in the Product-Seller tree. If this is a GROUP Forecast, then this returns the Forecast for the `product_group.root`. Similarly, if this is an INDIVIDUAL Forecast, then this returns the Forecast for the Product in the Seller that defined the `Product_Root`.
Properties: Export-Only Field

member_forecasts - - *a List(Forecast)/field of model Forecast*
A List of the Forecasts of this Forecast's Seller's 'members'.
Properties: Export-Only Field

active_member_forecasts - - *a List(Forecast)/field of model Forecast*
A List of the 'active' Forecasts of this Forecast's Seller's 'members'.
Properties: Export-Only Field

generic_forecasts - - *a List(Forecast)/field of model Forecast*
A List of the Forecasts that correspond to this Product's 'generic_products'. These are the Forecasts that represent the more generic demand being forecasted and allocated. This Forecast is essentially a subset of those more generic Forecasts. This Forecast will appear in these Forecast's `specific_forecasts`.
Properties: Export-Only Field

active_generic_forecasts - - *a List(Forecast)/field of model Forecast*
A List of the 'active' Forecasts that correspond to this Product's 'generic_products'.
Properties: Export-Only Field

specific_forecasts - - *a List(Forecast)/field of model Forecast*
A List of the Forecasts that correspond to this Product's `specific_products`. These are the Forecasts that represent the more specific demand being forecasted and allocated. This Forecast is generically covering all of the `specific_products`, and thus will be at least the sum of those. See the various fields of `Forecast_Entry` prefixed with `specifics_`. This Forecast will appear in these Forecast's `generic_forecasts`.
Properties: Export-Only Field

active_specific_forecasts - - *a List(Forecast)/field of model Forecast*
A List of the "active" Forecasts that correspond to this Product's `specific_products`.
Properties: Export-Only Field

request - - *a Request/field of model Forecast*
The Forecast Request corresponding to this Forecast. If there is none, create one.
There is no Forecast Request for either Group or Inactive Forecasts
Properties: Export-Only Field

entries - - *a list of Forecast_Entry submodels of model Forecast*
In each Forecast, there is one `Forecast_Entry` for each `Date_Range` in the `Seller_Plan's` 'forecast_horizon'. Each `Forecast_Entry` has the `Date_Range` it is forecasting, a raw 'forecasted_value', a 'committed' forecast value, an 'allocated' value that it can promise, a 'consumed' value that it has promised, and thus an 'gap' value that is still available to promise. It also contains fields that aggregate up those same values from this Seller's 'members', such as 'members_forecasted', 'members_committed', and 'members_allocated'.

Model	Forecast Model
-------	----------------

entries_consummed - - *a List(Quantity) field of model Forecast*
The total consumed quantity for each of the entries. (See Forecast_Entry.consumed)
Properties: Export-Only Field

entries_members_consummed - - *a List(Quantity) field of model Forecast*
The total Quantity of this Product for which actual Promises have been made by the members of this Seller for each of the entries. (See Forecast_Entry.members_consummed)
Properties: Export-Only Field

accept_as_allocated - - *a Void field of model Forecast*
Accept the allocated values in all of the Forecast_Entry's of this Forecast and then 'accept_as_allocated' any specific_forecasts and any 'member_forecasts'. This does 'accept_as_allocated' for each of the 'entries', which copies the allocated and 'retain_from_allocated' fields to the 'accepted' and 'retain_from_accepted' fields in the Forecast_Entry and in the corresponding Forecast_Entry(s) in every Forecast in that specific and member trees below it. It also sets 'date_accepted' in all of those Forecast_Entry's to 'now'.

Note that the full hierarchy is done because it is in general a bad idea to mix one set of allocations with another -- they may be completely inconsistent. However, you can always set any or all of the fields directly.

Properties: command=True Export-Only Field

allocate_allocated_available - - *a Void field of model Forecast*

This command allocates any quantity in 'allocated_available' of each Forecast_Entry to the Forecast_Entry's of member sellers

Properties: command=True Export-Only Field

owner - - *a Seller_Plan field of model Forecast*

Properties: Export-Only Field

product - - *a Product field of model INDIVIDUAL*

The Product for which demand this forecasts.

Properties: standard=True Export-Only Field

product_group - - *a Product_Group field of model GROUP*

The Product for which demand this forecasts.

Properties: standard=True Export-Only Field

Models	Forecast Model
--------	----------------

sub_forecasts - - *a List(Forecast) field of model GROUP*
The Forecasts for the 'sub_groups' and 'products' of this Forecast's 'product_group'.
Properties: standard=True Export-Only Field

active_sub_forecasts - - *a List(Forecast) field of model GROUP*
The Forecasts for the 'sub_groups' and active 'products' of this Forecast's 'product_group'.
Properties: standard=True Export-Only Field

active_leaf_product_forecasts - - *a List(Forecast) field of model GROUP*
The Forecasts for the active 'products' of this Forecast's 'product_group'. Returns all active descendants of this GROUP Forecast that are INDIVIDUAL forecasts. In other terms, these are the active "leaf" Forecasts of this product sub-hierarchy.
Properties: standard=True Export-Only Field

Models	Standard Extensions of model Forecast
--------	---------------------------------------

5.1.3.2.1.1 Standard Extensions of model Forecast
5.1.3.2.1.1.1 level extensions of model Forecast
5.1.3.2.1.1.1 INDIVIDUAL -- a level extension of model Forecast

An INDIVIDUAL Forecast is for an individual Product. The product defines how this Forecast is converted into Requests upon this Site.

The INDIVIDUAL model has these submodels :
ATP_Entry.

The INDIVIDUAL model has fields that references these models :
Product, ATP_Entry.

product -- a Product field of model INDIVIDUAL
The Product for which demand this forecast.
Properties: standard=True Export-Only Field

atp_entries -- a list of ATP_Entry submodels of model INDIVIDUAL.
In each Forecast, there is one ATP_Entry for each Date_Range in the Seller_Plan's list of Date_Ranges, 'atp_horizon'. Each ATP_Entry has the Date_Range for which it maintains allocations, an 'allocated' value that it can promise, a 'consumed' value that maintains the amount already promised, and an 'atp' value that is still available to promise.

5.1.3.2.1.1.2

GROUP -- a level extension of model Forecast

A GROUP Forecast is for a Product_Group, an aggregation of several Products' Forecasts. The product_group defines how this Forecast is propagated down to the Product_Group's sub_groups and products.

The GROUP model has fields that references these models :
Product_Group.

product_group -- a Product_Group field of model GROUP
The Product for which demand this forecast.

Properties: standard=True Export-Only Field

sub_forecasts -- a List(Forecast) field of model GROUP
The Forecasts for the sub_groups and products of this Forecast's product_group.
Properties: standard=True Export-Only Field

Models	GROUP Extension
--------	-----------------

active_sub_forecasts -- a List(Forecast) field of model GROUP

The Forecasts for the sub_groups and active products of this Forecast's product_group.

Properties: standard=True Export-Only Field

active_leaf_product_forecasts -- a List(Forecast) field of model GROUP

The Forecasts for the active products of this Forecast's product_group. Returns all active descendants of this GROUP Forecast that are INDIVIDUAL forecasts. In other terms, these are the active "leaf" Forecasts of this product sub-hierarchy.

Properties: standard=True Export-Only Field

use_std_split -- a Logical field of model GROUP

This field is obsolete. You must now set the use_std_split field in the Product_Group. Note that setting the field at the Product_Group is not exactly equivalent. The split

now applies to all Forecasts of the Product_Group.
Properties: obsolete=True Export-Only Field

5.1.3.2.1.2 Forecast_Entry Model

Forecast_Entry -- *a submodel of model Forecast*

In each Forecast, there is one Forecast_Entry for each Date_Range in the Seller_Plan's 'forecast_horizon'. Each Forecast_Entry has the Date_Range it is forecasting, a raw 'forecasted' value, a 'committed' 'forecast' value, an 'allocated' value that it can promise, a 'consumed' value that it has promised, and thus an 'airp' value that is still available to promise. It also contains fields that aggregate up those same values from this Seller's 'members', such as 'members_forecasted', 'members_committed', and 'members_allocated'.

The model has selectors:

forecast_policy, allocation_policy.

The Forecast_Entry model has fields that references these models :

Forecast.

These models have a field that is a Forecast_Entry model :

Forecast, Delivery_Request, Product_Available_To_Promise, NEGATIVE_ATP, NEGATIVE_PLANNED_ATP, OVER_COMMITTED, OVER_CONSUMED, UNALLOCATED_FORECAST.

The key field for this model is **delivery_dates**

This model may be extended with user-defined fields.

delivery_dates -- *a Date_Range field of model Forecast_Entry*

The delivery Dates forecasted by this Forecast_Entry. This cannot be set -- it is determined by the Seller_Plan's 'forecast_horizon'. All forecasting must be done for the same Date_Ranges for them to be comparable and aggregable.

This field is not really settable (to be read-only soon).

Default: None -- this is a key field

Properties: Export-Only Field

date_forecasted -- *a Date field of model Forecast_Entry*

The Date that the 'forecasted' value was last set. Setting 'forecasted' will set 'date_forecasted' to 'max(date_forecasted, now)'. As 'date_forecasted' is only for informational purposes, no problems are detected relative to it.

Default: the Date when 'forecasted' was set (initially infinite past)

forecasted -- *a Quantity field of model Forecast_Entry*

The Quantity of this 'product' or 'product_group' that the Seller believes can be sold for these 'delivery_dates'. This is the market potential. This may be an aggressive forecast -- it is not a commitment (see 'committed'). Rather, it is an upper bound on what can be 'committed'. Setting 'committed' higher than 'forecasted' results in an **OVER_COMMITTED** Problem.

Note that the default, "oo", implies that the market can support any level of sales. Thus, any 'committed' Forecast is reasonable. This makes this field essentially irrelevant -- the market potential is not a limit. By default, then, users can ignore this field and just set the 'committed' Forecast values.

If 'override_members_forecasted' is "true", then changes in 'members_forecasted' will have no effect on this 'forecasted' Quantity. This allows an 'organization' forecast to override what the 'members' are forecasting. For example, the 'members_forecasted' may be 600, but this Seller may believe that the total that will be sold during this period will be 800, independent of what the 'members' estimate. If the 'members' change their estimates to 700, this Seller still wants his 'forecasted' to remain 800.

In contrast, if 'override_members_forecasted' is "false", then any increase or decrease in 'members_forecasted' will increase or decrease this 'forecasted' Quantity by the same Quantity. This allows an 'organization' to add a certain Quantity either for its own sales or as an estimate of how much the 'members' typically under or over forecast. Thus, in the previous example, when this Seller set 'forecasted' to 800, the Seller may have meant "I want my 'forecasted' to be 200 more than my 'members' estimate; so, when the 'members' sum increases to 700, this 'forecasted' should rise to 900 (remaining 200 more)".

Note that 'forecasted' cannot be set to less than 'specifics_forecasted', though typically 'forecasted' is significantly more than 'specifics_forecasted' anyway. If this Forecast has 'specific_forecasts' (if its 'product' is the Generic_Product of another Product), then this 'forecasted' Quantity includes all of the 'forecasted' Quantities of its 'specific_forecasts'. Thus, if 'specifics_forecasted' is ever increased to larger than 'forecasted', then 'forecasted' will be increased to match.

This Quantity is converted to the 'unit' of the 'product' or 'product_group'.

Default: oo

cumulative_forecasted -- *a Quantity field of model Forecast_Entry*

The sum of 'forecasted' for each Forecast_Entry from the first through this one.

This Quantity is converted to the 'unit' of the 'product' or 'product_group'.

Models	Forecast_Entry Model
--------	----------------------

Properties: Export-Only Field

specifics_forecasted - - *a Quantity field of model Forecast_Entry*
The sum of the 'forecasted' quantities for these 'delivery_dates' for the 'specific_forecasts' (the 'Forecast' for this 'Product's 'specific_products'). This will be zero if this is a GROUP Forecast or if this 'Product' is not a 'Generic_Product' (if it has no 'specific_products').

Note that 'forecasted' cannot be set to less than 'specifics_forecasted', and that the 'forecasted' value will be automatically increased such that it is always at least as much as 'specifics_forecasted', since the 'specific_forecast' are essentially the subsets of the 'generic_demand_modelled' by this 'Forecast'. Typically, however, 'forecasted' is more than 'specifics_forecasted' due to the uncertainty in the demand for 'specific_products'.

This Quantity is converted to the 'unit' of the 'product'.

Properties: Export-Only Field

members_forecasted - - *a Quantity field of model Forecast_Entry*
The sum of the 'forecasted' quantities for these 'delivery_dates' for this 'Product' by the 'member_forecasts' (the 'Forecast' of the 'Sellers' that are 'members' of this 'Seller'). This will be zero if this 'Seller' has no 'members'.

If 'forecasted' is less (more) than this, then the 'Seller' is asserting that the 'members' are overly optimistic (pessimistic).

If 'override_members_forecasted' is "true", then changes in 'members_forecasted' will have no effect on this 'forecasted' Quantity. In contrast, if 'override_members_forecasted' is "false", then any increase or decrease in 'members_forecasted' will increase or decrease this 'forecasted' Quantity by the same Quantity. See the further discussion and examples discussed in the 'forecasted' field.

This field can be set, which results in proportional increase or decrease of all of the corresponding 'forecasted' fields of the 'member_forecasts' which make up this sum. So, consider the case that there are four 'member_forecasts' with corresponding 'forecasted' quantities of 300, 200, 100, and 0, making this field 600; if you set this field to 300, then the 'forecasted' fields of the 'member_forecasts' will be set to 150, 100, 50, and 0, respectively. If this field is set to 0, then 'forecasted' fields of the 'member_forecasts' will be set to 0. Subsequently, if this value is set to 600, this value will be equally distributed among the members, i.e. the 'forecasted' fields of the all 'member_forecasts' will be set to 150.

This Quantity is converted to the 'unit' of the 'product' or 'product_group'.

Models	Forecast_Entry Model
--------	----------------------

Default: sum of 'forecasted' for all 'members'

override_members_forecasted - - *a Logical field of model Forecast_Entry*
If 'override_members_forecasted' is "true", then changes in 'members_forecasted' will have no effect on this 'forecasted' Quantity. In contrast, if 'override_members_forecasted' is "false", then any increase or decrease in 'members_forecasted' will increase or decrease this 'forecasted' Quantity by the same Quantity. See the further discussion and examples discussed in the 'forecasted' field.

If this is an INDIVIDUAL Forecast, then if 'prod-uct.always_override_members_forecasted' is "true", then this value is "true" (and cannot be set to "false"). If this is a GROUP Forecast, then this value is "true" (and cannot be set to "false"). Otherwise, this field may be set either "true" or "false".

Currently, this field is not settable; you can only set 'prod-uct.always_override_members_forecasted'.

Properties: Export-Only Field

date_committed - - *a Date field of model Forecast_Entry*
The Date that the 'committed' value was last set (requested). Setting 'committed' will set 'date_committed' to 'max(date_committed, now)'. If 'date_committed' is later than 'date_allocated', then an UNALLOCATED_FORECAST Problem will be created to indicate that a different committed forecast has been created but not yet satisfied or rejected.

Default: the Date when 'committed' was set (initially infinite past)

committed - - *a Quantity field of model Forecast_Entry*
The Quantity of this 'product' or 'product_group' that this 'Seller' is willing to commit to selling for these 'delivery_dates'. This could also be called "requested allocation". It is the Quantity that will be allocated to this particular 'Seller' if there is sufficient supply.

Typically, the 'Seller' organization will set this higher than the 'members_committed', committing that several of the 'members' will sell more than they committed (though which of the 'members' will do so may be anyone's guess). In this way, additional 'available-to-promise' will be allocated to the organization, available to the 'members' on a first-come-first-served basis. So, as members sell more than they committed, and thus lack 'allocated' 'available-to-promise', they can begin to consume from their organization's ATP.

Note that 'committed' is not synonymous with ATP. The 'committed' value results in forecast Requests. If those Requests can be met and promises are made from the supplier Slices, then those Promises constitute the allocated Quantity, the unconsumed portion of which is ATP. Those Promises have been allocated to the Seller, and the Seller can use those Promises to immediately make Promises to actual Requests from customer Slices.

If this is set larger than 'forecasted' (the estimated market demand), an OVER_COMMITTED Problem will be raised. If a separate 'forecasted' value is not desired, set all 'forecasted' values to "00" (the default) which will effectively make those limits vanish.

If 'override_members_committed' is "true", then changes in 'members_committed' will have no effect on this 'committed' Quantity. This allows an 'organization' commitment to override what the 'members' are committing. For example, the 'members_committed' may be 600, but this Seller may believe that the total that can be sold during this period will be 800, independent of what the 'members' estimate. If the 'members' change their estimates to 700, this Seller still wants his 'committed' to remain 800.

In contrast, if 'override_members_committed' is "false", then any increase or decrease in 'members_committed' will increase or decrease this 'committed' Quantity by the same Quantity. This allows an 'organization' to add a certain Quantity either for its own sales or as an estimate of how much the 'members' typically under or over forecast. Thus, in the previous example, when this Seller set 'committed' to 800, the Seller may have meant "I want my 'committed' to be 200 more than my 'members' commitment; so, when the 'members' sum increases to 700, this 'committed' should rise to 900 (remaining 200 more).

Note that 'committed' cannot be set to less than 'specifics_committed', though typically 'committed' is significantly more than 'specifics_committed' anyway. If this Forecast has 'specific_forecast' (if its product is the Generic_Product of another Product), then this 'committed' Quantity includes all of the 'committed' Quantities of its 'specific_forecast'. Thus, if 'specifics_committed' is ever increased to larger than 'committed', then 'committed' will be increased to match.

This Quantity is converted to the 'unit' of the product or product_group.
Default: 0

cumulative_committed - - a Quantity field of model Forecast_Entry
The sum of 'committed' for each Forecast_Entry from the first through this one.

This Quantity is converted to the 'unit' of the product or product_group.
Properties: Export-Only Field

specifics_committed - - a Quantity field of model Forecast_Entry
The sum of the 'committed' quantities for these 'delivery_dates' for the 'specific_forecast' (the Forecast for this Product's 'specific_products'). This will be zero if this is a GROUP Forecast or if this Product is not a Generic_Product (if it has no 'specific_products').

Note that 'committed' cannot be set to less than 'specific_committed', and that the 'committed' value will be automatically increased such that it is always at least as much as 'specifics_committed', since the 'specific_forecasts' are essentially the subsets of the generic demand modelled by this Forecast. Typically, however, 'committed' is more than 'specifics_committed' due to the uncertainty in the demand for specific products.

Further, note that the forecast requests generated by this Forecast are only for the delta between 'committed' and 'specifics_committed'. The 'specifics_committed' Quantity is covered by the forecast requests generated by the 'specific_forecasts'. Of course, also subtracted out is the Quantity for any actual Requests or Promises that are planned, which are typically all on the 'specific_forecasts'.

This Quantity is converted to the 'unit' of the product:

Properties: Export-Only Field

members_committed - - a Quantity field of model Forecast_Entry
The sum of the 'committed' quantities for these 'delivery_dates' for this Product by the 'member_forecasts' (the Forecasts of the Sellers that are members of this Seller). This will be zero if this Seller has no 'members'.

If 'committed' is less (more) than this, then the Seller is asserting that the 'members' are overly optimistic (pessimistic).

If 'override_members_committed' is "true", then changes in 'members_committed' will have no effect on this 'committed' Quantity. In contrast, if 'override_members_committed' is "false", then any increase or decrease in 'members_committed' will increase or decrease this 'committed' Quantity by the same Quantity. See the further discussion and examples discussed in the 'committed' field.

Models**Forecast_Entry Model**

This field can be set, which results in proportional increase or decrease of all of the corresponding 'committed' fields of the 'member_forecast' which make up this sum. So, consider the case that there are four 'member_forecast' with corresponding 'committed' quantities of 300, 200, 100, and 0, making this field 600; if you set this field to 300, then the 'committed' fields of the 'member_forecast' will be set to 150, 100, 50, and 0, respectively. If this field is set to 0, then 'committed' fields of the 'member_forecast' will be set to 0. Subsequently, if this value is set to 600, this value will be equally distributed among the members, i.e. the 'committed' fields of the all 'member_forecast' will be set to 150.

This Quantity is converted to the 'unit' of the 'product' or 'product_group'.
Default: sum of 'committed' for all members

override_members_committed - - *a Logical field of model Forecast_Entry*
If 'override_members_committed' is "true", then changes in 'members_committed' will have no effect on this 'committed' Quantity. In contrast, if 'override_members_committed' is "false", then any increase or decrease in 'members_committed' will increase or decrease this 'committed' Quantity by the same Quantity. See the further discussion and examples discussed in the 'committed' field.

If this is an **INDIVIDUAL** Forecast, then if 'product_always_override_members_committed' is "true", then this value is "true" (and cannot be set to "false"). If this is a **GROUP** Forecast, then this value is "true" (and cannot be set to "false"). Otherwise, this field may be set either "true" or "false".

Currently, this field is not settable; you can only set 'product_always_override_members_committed'.

Properties: Export-Only Field

retain_from_allocated - - *a Quantity field of model Forecast_Entry*

The Quantity of the 'allocated' (and 'planned') that should be retained at this Seller. This Quantity is 'available' from this Seller, but will not be allocated down to the members not available for use by the members first-come-first-served.

This field, though computed by the 'allocation_policy' and related fields of the Product, is settable. Setting 'retain' will also set 'lock', 'retain_from_allocated' to "true" -- which will prevent this field from being recomputed by the 'allocation_policy' of the Product -- it will remain at the Quantity set into it.

This allows the user to easily manipulate the 'retain_from_allocated' value for a particular bucket, without needing to modify the Product parameters. This is often used to "release" some of the retained Quantity when more is needed by the members:

Models**Forecast_Entry Model**

Default: computed from the Product fields

lock_retain_from_allocated - - *a Logical field of model Forecast_Entry*
If "false", then the 'retain' Quantity is computed from the Product 'allocation_policy' and related fields. If "true", then the current 'retain' Quantity will be used, overriding the setting that would normally be computed based upon the fields in the Product.

This allows the user to easily manipulate the 'retain' value for a particular bucket, without needing to modify the Product parameters. This is often used to "release" some of the 'retained' Quantity when more is needed by the members:

Note that setting this to "false" will cause the 'retain' Quantity, and therefore the 'retained' Quantity, to be recomputed from the Product fields.
Default: false

retain_from_accepted - - *a Quantity field of model Forecast_Entry*

The Quantity of the 'accepted' that should be retained at this Seller. This Quantity is 'available' from this Seller, but will not be available to the members (as first-come-first-served ATP).

This field is set from 'retain_from_allocated' when 'accept_as_allocated' is called. This field can also be set directly. This allows the user to easily manipulate the retained value for a particular bucket, which is often used to "release" some of the retained Quantity when more is needed by the members:
Default: 0

planned - - *a Quantity field of model Forecast_Entry*

The Quantity of this 'product' or 'product_group' that is currently planned to be delivered. This is an "unpromised" or "what-if" variation of 'allocated'. It is allocated to the members using the same 'allocation_policy' as 'allocated'. If current plans are not satisfying the Promises that make up 'allocated', then this value may be less than 'allocated'. On the other hand, if a planning effort is underway to increase 'allocated', this value may be larger than 'allocated'. In a sense, this often represents what 'allocated' will be soon -- the "what if I planned like this, what would be my 'allocated' value?"

This Quantity is converted to the 'unit' of the 'product' or 'product_group'.
Properties: Export-Only Field

cumulative_planned - - *a Quantity field of model Forecast_Entry*
The sum of 'planned' for each Forecast_Entry from the first through this one.

This Quantity is converted to the 'unit' of the 'product' or 'product_group'.

Models	Forecast_Entry Model
--------	----------------------

Properties: Export-Only Field

specifics_planned - - *a Quantity field of model Forecast_Entry*
The sum of the 'planned' quantities for these 'delivery_dates' for the 'specific_forecasts' (the 'Forecast' for this Product's 'specific_products'). This will be zero if this is a 'GROUP Forecast' or if this Product is not a 'Generic_Product' (if it has no 'specific_products').

Note that 'planned' will never be less than 'specifics_planned', since the 'specific_forecasts' are essentially the subsets of the generic demand modelled by this Forecast. Typically, however, 'planned' is more than 'specifics_planned' due to the uncertainty in the demand for specific products.

This Quantity is converted to the 'unit' of the 'product'.

Properties: Export-Only Field

members_planned - - *a Quantity field of model Forecast_Entry*
The sum of the 'planned' quantities for these 'delivery_dates' for this Product by the 'member_forecasts' (the 'Forecasts' of the 'Sellers' that are 'members' of this 'Seller'). This will be zero if this 'Seller' has no 'members'.

This Quantity is converted to the 'unit' of the 'product' or 'product_group'.
Properties: Export-Only Field

date_allocated - - *a Date field of model Forecast_Entry*
The Date that the 'allocated' value was last set (promised). Setting 'allocated' will set 'date_allocated' to 'max(date_allocated, now)'.

If 'date_committed' is later than 'date_allocated', then a 'FORECAST_NOT_ALLOCATED' (an 'UNALLOCATED_FORECAST') Problem will be created to indicate that a different commitment has been created but not yet satisfied or rejected. Rejecting it is as simple as setting this field to 'now'.

If 'date_allocated' is later than 'date_accepted', then an 'ALLOCATION_NOT_ACCEPTED' Problem will be created to indicate that a different allocation has been made, but has not yet been accepted or rejected. Rejecting it is as simple as setting 'date_accepted' to 'now'. Accepting it can be done by 'accept_as_allocated'.

Default: the Date when 'allocated' was set (initially infinite past)

Models	Forecast_Entry Model
--------	----------------------

allocated - - *a Quantity field of model Forecast_Entry*
The Quantity of this 'product' or 'product_group' for which forecast Promises have been allocated to the top-seller for these 'available_dates'. This could also be called "preliminary gross ATP" - - the ATP prior to being consumed by actual Promises. The ATP is preliminary since this 'allocated' Quantity may not yet be 'accepted'. The 'allocated_available' is this 'allocated' minus the 'consumed' (the actual Promises that consume from this 'Forecast_Entry').

If 'allocated' is less than 'committed', then additional allocation may be obtained from the 'Seller's' 'organization', or if this 'Seller' defines the 'Product_Root', then from the supplier Slices.

This Quantity is converted to the 'unit' of the 'product' or 'product_group'.

Default: 0

lock_allocated - - *a Logical field of model Forecast_Entry*

If 'lock_allocated' is "true", then the 'organization's' 'allocation_policy' will not adjust this 'allocated' Quantity. If 'lock_allocated' is "false", then this 'allocated' Quantity will be adjusted according to the 'organization's' 'allocation_policy'. See the 'allocated' field and the 'Product_allocation_policy' field for further discussion.

Default: false

cumulative_allocated - - *a Quantity field of model Forecast_Entry*
The sum of 'allocated' for each 'Forecast_Entry' from the first through this one.

This Quantity is converted to the 'unit' of the 'product' or 'product_group'.
Properties: Export-Only Field

specifics_allocated - - *a Quantity field of model Forecast_Entry*
The sum of the 'allocated' quantities for these 'delivery_dates' for the 'specific_forecasts' (the 'Forecasts' for this Product's 'specific_products'). This will be zero if this is a 'GROUP Forecast' or if this Product is not a 'Generic_Product' (if it has no 'specific_products').

Note that 'allocated' cannot be set to less than 'specifics_allocated', and that the 'allocated' value will be automatically increased such that it is always at least as much as 'specifics_allocated', since the 'specific_forecasts' are essentially the subsets of the generic demand modelled by this Forecast. Typically, however, 'allocated' is more than 'specifics_allocated' due to the uncertainty in the demand for specific products.

This Quantity is converted to the 'unit' of the 'product'.

Properties: Export-Only Field

Models Forecast_Entry Model

members_allocated -- *a Quantity field of model Forecast_Entry*
The sum of the 'allocated' quantities for these 'delivery_dates' for this Product by the 'member_forecasts' (the Forecasts of the Sellers that are members of this Seller). This will be zero if this Seller has no 'members'.

This Quantity is converted to the 'unit' of the 'product' or 'product_group'.
Properties: Export-Only Field

date_accepted -- *a Date field of model Forecast_Entry*
The Date that the 'accepted' value was last set. Setting 'accepted' will set 'date_accepted' to 'max(date_accepted, now)'.

If 'date_allocated' is later than 'date_accepted', then an **ALLOCATION_NOT_ACCEPTED** Problem will be created to indicate that a different allocation has been made, but has not yet been accepted or rejected. Rejecting it is as simple as setting 'date_accepted' to 'now'. Accepting it can be done by 'accept_as_allocated'.
Default: the Date when 'accepted' was set (initially infinite past)

accepted -- *a Quantity field of model Forecast_Entry*
The Quantity of this 'product' or 'product_group' for which Promises have been allocated to this Seller for these 'delivery_dates'. This could also be called 'gross ATP' -- the ATP prior to being consumed by actual Promises. The available 'to_promise' is this 'accepted' minus the 'consumed' (the actual Promises that consume from this Forecast_Entry).

If 'allocated' is less than 'committed', then additional allocation may be obtained from the Seller's organization, or if this Seller defines the Product_Root, then from the supplier Sites.

This Quantity is converted to the 'unit' of the 'product' or 'product_group'.
Default: 0

cumulative_accepted -- *a Quantity field of model Forecast_Entry*
The sum of 'accepted' for each Forecast_Entry from the first through this one.

This Quantity is converted to the 'unit' of the 'product' or 'product_group'.
Properties: Export-Only Field

Models Forecast_Entry Model

members_accepted -- *a Quantity field of model Forecast_Entry*
The sum of the 'accepted' quantities for these 'delivery_dates' for this Product by the 'member_forecasts' (the Forecasts of the Sellers that are members of this Seller). This will be zero if this Seller has no 'members'.

This Quantity is converted to the 'unit' of the 'product' or 'product_group'.
Properties: Export-Only Field

accept_as_allocated -- *a Void field of model Forecast_Entry*
Accept the allocated values in this Forecast_Entry and then 'accept_as_allocated' in the corresponding Forecast_Entry of any specific 'forecasts', and similarly 'accept_as_allocated' the corresponding Forecast_Entry(s) of any 'member_forecasts'. This copies the 'allocated' and 'retain_from_allocated' fields to the 'accepted' and 'retain_from_accepted' fields in this Forecast_Entry and every Forecast_Entry below it in the specific and member trees below it. It also sets 'date_accepted' in all of those models to 'now'.

Note that the full hierarchy is done because it is in general a bad idea to mix one set of allocations with another -- they may be completely inconsistent. However, you can always set any or all of the fields directly.

Properties: command=True Export-Only Field

consumed -- *a Quantity field of model Forecast_Entry*
The total Quantity of this Product for which actual Promises have been made for these 'delivery_dates', consuming this Forecast_Entry's 'accepted' allocation.

This Quantity is converted to the 'unit' of the 'product' or 'product_group'.
Properties: Export-Only Field

cumulative_consumed -- *a Quantity field of model Forecast_Entry*
The sum of 'consumed' for each Forecast_Entry from the first through this one.

This Quantity is converted to the 'unit' of the 'product' or 'product_group'.
Properties: Export-Only Field

specifics_consumed -- *a Quantity field of model Forecast_Entry*
The sum of the 'consumed' quantities for these 'delivery_dates' for the 'specific_forecasts' (the Forecasts for this Product's specific_products). This will be zero if this is a GROUP Forecast or if this Product is not a Generic_Product (if it has no 'specific_products').

Note that 'consumed' includes the 'specifics_consumed'. Each actual promise made whose 'consumed_forecast' is one of the 'specific_forecasts' will increase 'consumed' not only of that specific Forecast, but also for each of its generic Forecasts (and each of their generic Forecasts, and so on).

This Quantity is converted to the unit of the product:

Properties: Export-Only Field

members_consumed -- *a Quantity field of model Forecast_Entry*
The total Quantity of this Product for which actual Promises have been made for these 'delivery_dates' by the 'members' of this Seller. The sum of the 'consumed' quantities for these 'delivery_dates' for this Product from the 'member_forecasts' (the Forecasts of the Sellers that are 'members' of this Seller). This will be zero if this Seller has no 'members'.

This Quantity is converted to the unit of the product or 'product_group':

Properties: Export-Only Field

actual_promises -- *a ListItem_Promise field of model Forecast_Entry*
The actual (customer) Item_Promises that have consumed the allocated Quantity of this Forecast_Entry. The sum of these are added to the combination (not necessarily sum) of the 'specifics_consumed' and the 'members_consumed' and to form the 'consumed' of this Forecast_Entry. (Note that the sum of 'specifics_consumed' and 'members_consumed' may double-count the actual_promises of the 'specific_forecasts' within the member Seller_Plans.)

Properties: Export-Only Field

available_to_promise -- *a Quantity field of model Forecast_Entry*
This is 'accepted - consumed'. It is the total Quantity of this product or 'product_group' that is planned to be built, allocated to this Seller, and unconsumed, such that it is available to the Seller to be promised to customers for these 'delivery_dates'.

This Quantity is converted to the unit of the product or 'product_group':

The field 'available_to_promise' is an alias of 'available' -- provided to match the commonly used term -- but 'available' is more consistent with the family of fields.

Properties: Export-Only Field

available -- *a Quantity field of model Forecast_Entry*
This is 'accepted - consumed'. It is the total Quantity of this product or 'product_group' that is planned to be built, allocated to this Seller, and unconsumed, such that it is available to the Seller to be promised to customers for these 'delivery_dates'.

This Quantity is converted to the unit of the product or 'product_group':

The field 'available_to_promise' is an alias of 'available' -- provided to match the commonly used term -- but 'available' is more consistent with the family of fields.

Properties: Export-Only Field

cumulative_available -- *a Quantity field of model Forecast_Entry*
The sum of 'available_to_promise' for each Forecast_Entry from the first through this one.

This is the cumulative Quantity of this product or 'product_group' over all previous Forecast_Entries that is planned to be built, allocated to this Seller, and unconsumed, such that it is available to the Seller to be promised to customers for these 'delivery_dates'.

This Quantity is converted to the unit of the product or 'product_group':

Properties: Export-Only Field

members_available -- *a Quantity field of model Forecast_Entry*
The sum of the 'available_to_promise' quantities for these 'delivery_dates' for this Product by the 'member_forecasts' (the Forecasts of the Sellers that are 'members' of this Seller). This will be zero if this Seller has no 'members'.

This Quantity is converted to the unit of the product or 'product_group':

Properties: Export-Only Field

specifics_available -- *a Quantity field of model Forecast_Entry*
The sum of the 'available_to_promise' quantities for these 'delivery_dates' for the 'specific_forecasts' (the Forecasts for this Product's 'specific_products'). This will be zero if this is a GROUP Forecast or if this Product is not a Generic_Product (if it has no 'specific_products').

This Quantity is converted to the unit of the product:

Properties: Export-Only Field

allocated_available - - *a Quantity field of model Forecast_Entry*

This is simply 'allocated - consumed'. It is the total Quantity of this 'product' or 'product_group' that would be available to the Seller to be promised to customers for these 'delivery_dates' if the currently 'allocated' quantities were accepted. In a sense, this is the "what-if" available_to_promise.

This Quantity is converted to the 'unit' of the 'product' or 'product_group'.

Properties: Export-Only Field

cumulative_allocated_available - - *a Quantity field of model Forecast_Entry*
The sum of 'allocated_available' for each Forecast_Entry from the first through this one.

This is the cumulative Quantity of this 'product' or 'product_group' over all previous Forecast_Entries would be available to the Seller to be promised to customers for these 'delivery_dates' if the currently 'allocated' quantities were accepted. In a sense, this is the "what-if" cumulative_available.

This Quantity is converted to the 'unit' of the 'product' or 'product_group'.

Properties: Export-Only Field

planned_available - - *a Quantity field of model Forecast_Entry*

This is simply 'planned - consumed'. It is the total Quantity of this 'product' or 'product_group' that would be available to the Seller to be promised to customers for these 'delivery_dates' if the currently 'planned' allocations were promised and accepted. In a sense, this is the "what-if" available_to_promise.

This Quantity is converted to the 'unit' of the 'product' or 'product_group'.

Properties: Export-Only Field

cumulative_planned_available - - *a Quantity field of model Forecast_Entry*

The sum of 'planned_available' for each Forecast_Entry from the first through this one.

This is the cumulative Quantity of this 'product' or 'product_group' over all previous Forecast_Entries would be available to the Seller to be promised to customers for these 'delivery_dates' if the currently 'planned' allocations were promised and accepted. In a sense, this is the "what-if" cumulative_available.

This Quantity is converted to the 'unit' of the 'product' or 'product_group'.

Properties: Export-Only Field

zero_available_to_promise - - *a Logical field of model Forecast_Entry*

If 'zero_available_to_promise' is "true", then this Forecast_Entry will never allow 'available_to_promise' to be greater than zero. If a situation which would cause 'available_to_promise' to be greater than zero occurs, this Forecast_Entry will automatically release any 'allocated' quantities to its 'organization' in order to force 'available_to_promise' to be zero. If 'zero_available_to_promise' is "false", then 'available_to_promise' will work normally for this Forecast_Entry.

Default: false

allocate_allocated_available - - *a Void field of model Forecast_Entry*

This command allocates any quantity in 'allocated_available' to the member sellers

Properties: command=True Export-Only Field

forecast_policy - - *an extension selector of model Forecast_Entry*

This is defined by the product's 'forecast_policy'. The 'forecast_policy' determines the additional fields that are forecasted per entry.

Default: SINGLE_REQUEST

Properties: Export-Only Field

allocation_policy - - *an extension selector of model Forecast_Entry*

This is defined by the product's 'allocation_policy'. The 'allocation_policy' determines the additional fields that are forecasted per entry.

Default: PER_COMMITTED

Extensions:

MEMBER_RANK.

forecast_requests - - *a List(Item.Request) field of model Forecast_Entry*

The forecast Item_Requests generated by this Forecast_Entry's committed value.

Note that this will only be populated for the Seller_Plan of the Seller that owns this Product. If this Product is owned by an 'organization' of this Seller, then this Seller will receive its 'allocated' value from its 'organization' and this list will be empty.

Properties: Export-Only Field

forecast_promises - - *a List(Item.Promise) field of model Forecast_Entry*

The forecast Item_Promises corresponding to 'forecast_requests'. If this Seller owns this Product, then the sum of these Item_Promises form 'allocated'. This is equivalent to 'forecast_requests' for each 'item_promise'.

Properties: Export-Only Field

owner - - *a Forecast field of model Forecast_Entry*

Module	Forecast_Entry Model
--------	----------------------

Properties: Export-Only Field

Module	Level submodels of model Forecast
--------	-----------------------------------

5.1.3.2.1.3 level submodels of model Forecast
5.1.3.2.1.4 ATP_Entry Model

ATP_Entry -- a submodel of model Forecast

In each Forecast, there is one ATP_Entry for each Date_Range in the Seller_Plan's list of Date_Ranges, 'ship_horizon'. Each ATP_Entry has the Date_Range for which it maintains allocations, an 'allocated' value that it can promise, a 'consumed' value that maintains the amount already promised, and an 'ship' value that is still available to promise.

The ATP_Entry model has fields that references these models :
Forecast.

These models have a field that is a ATP_Entry model :
INDIVIDUAL.

The key field for this model is available_dates

available_dates -- a Date_Range field of model ATP_Entry
The dates containing allocations in this ATP_Entry. This cannot be set -- it is determined by the Seller's 'ship_horizon' extension. All allocation must be done for the same Date_Ranges for them to be comparable and aggregable.

This field is not really settable (to be read-only soon).
Default: None -- this is a key field
Properties: Export-Only Field

allocated -- a Quantity field of model ATP_Entry
The Quantity of this 'product' for which forecast Promises have been allocated to the top-seller for these 'available_dates'. This could also be called 'preliminary gross ATP' -- the ATP prior to being consumed by actual Promises. The ATP is preliminary since the forecast Promises have not yet been accepted. The 'allocated available to promise' is this 'allocated' minus the actual Promises that have been made in a Forecast_Entry by consuming this allocation.

This Quantity is converted to the 'unit' of the 'product' or 'product_group'.
Default: 0

consumed -- a Quantity field of model ATP_Entry
The total Quantity of this Product for which actual Promises have been made for these 'available_dates', consuming this ATP_Entry's allocated quantity.

Models	ATP_Entry Model
--------	-----------------

This Quantity is converted to the unit of the product or product_group.
Default: 0

Properties: Export-Only Field

allocated_available -- *a Quantity field of model ATP_Entry*

This is simply 'allocated - consumed'. It is the total Quantity of this 'product' or 'product_group' that would be available to the Seller to be promised to customers for these 'available_dates'.

This Quantity is converted to the unit of the product or product_group.
Properties: Export-Only Field

cumulative_allocated_available -- *a Quantity field of model ATP_Entry*

This Quantity represents the cumulative allocated 'available' for these 'available_dates' that would be available to the Seller to be promised to customers for these 'available_dates'.

This Quantity is converted to the unit of the product or product_group.
Properties: Export-Only Field

owner -- *a Forecast field of model ATP_Entry*

Properties: Export-Only Field

Models	Problem Model
--------	---------------

5.1.3.3 Problem Model

Problem -- *a submodel of model Plan*

The Problem models a violation in a Plan of a Supply Chain. Problems include both feasibility problems (things that must be eliminated for the Plan to be valid: such as use of non-existent capacity) and desirability problems (things that would be preferable to eliminate, but not necessary to use the plan, such as late orders). A desirability problem in one company, may be a feasibility problem in another -- thus that division is user definable in many cases.

The model has selectors:
category.

The Problem model has fields that references these models:
Plan.

These models have a field that is a Problem model:
Plan, Delivery_Request, Delivery_Promise, Delivery_Acceptance, Active_Problem, Item_Request, Item_Acceptance, Item_Promise.

The key field for this model is description

description -- *a Computed_String field of model Problem*
A textual description of the Problem.

Default: None -- this is a key field
Properties: Export-Only Field

dates -- *a Date_Range field of model Problem*
The Date_Range over which this Problem occurs.

Properties: Export-Only Field

feasible -- *a Logical field of model Problem*

If "false", then this models a feasibility Problem -- the Plan is not feasible. If "true", then this models an undesirable, but feasible condition.

Properties: Export-Only Field

cost -- *a Money field of model Problem*

The cost incurred due to this Problem remaining. If 'feasible' is "false", then this 'cost' is infinite.

Properties: Export-Only Field

Model	Problem Model
-------	---------------

last_change -- a Date field of model Problem

The Date for which the last change was made for this problem. This field is used by the bookmark facility to determine which problems have been modified since the bookmark was established. It is settable, which can be useful when working with certain Strategies, Problem_Sets, or Problem-solving Reports.

Default: the Date that the Problem was created or last modified

category -- an extension selector of model Problem

The category of Problem. In addition to categorization, it also determines the fields that define what has the Problem and how severe the Problem is. For example, an OVERLOAD Problem will have a Resource_Plan (what is overloaded) and an 'excess_load' field which defines how much overload occurs during the 'dates'.

Note that these same extensions and fields appear in the Problem_Set submodel of Strategy for specifying which Problems the Strategy should be working on.

Properties: Export-Only Field

Extensions:

REQUEST_NOT_PLANNED, REQUEST_PLANNED_LATE,
REQUEST_PLANNED_EARLY, REQUEST_PLANNED_SHORT,
REQUEST_PLANNED_EXCESS, PROMISE_NOT_PLANNED,
PROMISE_PLANNED_LATE, PROMISE_PLANNED_EARLY,
PROMISE_PLANNED_SHORT, PROMISE_PLANNED_EXCESS,
ACCEPTANCE_NOT_PLANNED, ACCEPTANCE_PLANNED_LATE,
ACCEPTANCE_PLANNED_EARLY, ACCEPTANCE_PLANNED_SHORT,
ACCEPTANCE_PLANNED_EXCESS, PLANNED_BEFORE_CURRENT,
UNRELEASED_NEEDS_RELEASE, INCONSISTENT_OPPPLAN,
REQUEST_PROMISED_LATE, REQUEST_PROMISED_EARLY,
REQUEST_PROMISED_SHORT, REQUEST_PROMISED_EXCESS,
ITEM_PROMISE_OVERPRICED, DELIVERY_PROMISE_OVERPRICED,
PROMISE_NOT_OFFERED, PROMISE_NOT_ACCEPTED,
ACCEPTANCE_INCONSISTENT, REQUEST_QUEUED,
DELIVERY_REQUEST_NOT_COORDINATED,
DELIVERY_PROMISE_NOT_COORDINATED,
DELIVERY_ACCEPTANCE_NOT_COORDINATED, NEGATIVE_ATP,
NEGATIVE_PLANNED_ATP, OVER_COMMITTED, OVER_CONSUMED,
UNALLOCATED_FORECAST, SUPPLY_PLANNED_LATE,
SUPPLY_PLANNED_EARLY, SUPPLY_PLANNED_SHORT,
SUPPLY_PLANNED_EXCESS, SUPPLY_PROMISED_LATE,
SUPPLY_PROMISED_EARLY, SUPPLY_PROMISED_SHORT,
SUPPLY_PROMISED_EXCESS, UNIDENTIFIED_OF_STATE, UNCONSOLIDATED,
UNCOORDINATED, CONSOLIDATION_OVERSIZE,

Model	Problem Model
-------	---------------

CONSOLIDATION, UNDERSIZE, OVERLOAD, OVERSIZE,
BUCKET_OVERSIZE, UNDERLOAD, NEGATIVE_ON_HAND,
OVER_FLOW_LIMIT, NEGATIVE_ON_HAND_AT_END,
LOT_OVER_CONSUMED, LOT_NOT_CONSUMED,
LOT_NOT_PRODUCED, LOT_OVER_PRODUCED, LOW_ON_HAND,
EXCESS_ON_HAND, EXCESS_ON_HAND_AT_END,

in_category (Problem_Category) -- a Logical field of model Problem

Returns "true" if this Problem is in the specified Problem_Category.

Properties: Export-Only Field

interaction -- a Number field of model Problem

Heuristic estimate of the interaction of this Problem with other Problems and potential Problems. The higher this number, the more difficult it will likely be to resolve this Problem without creating other Problems.

Properties: Export-Only Field

resolvable -- a Logical field of model Problem

If "false", then this Problem has been determined to be unresolvable by the automated solvers -- human intervention will be needed.

Properties: Export-Only Field

resolve, resolve (Active_Strategy) -- a Void field of model Problem

Attempts to eliminate this Problem under the direction of the currently 'running' Active_Strategy of the owner Plan. If there is no Active_Strategy 'running', then it uses the 'auto_run' Active_Strategy. The resolve(Active_Strategy) function resolves a problem under the direction of the specified active strategy (whether it is 'running' or not).

Properties: command=True Export-Only Field

owner -- a Plan field of model Problem

Properties: Export-Only Field

5.1.3.4 Active_Strategy Model

Active_Strategy -- a submodel of model Plan

The Strategy's that remain active and up-to-date on every Plan change.

The Active_Strategy models an approach to resolving the remaining Problems in a Plan. It consists of a specification of what Problems to attempt to resolve, what modifications can be made in those attempts, and what criteria to use for judging the "goodness" of the Plan.

The model has selectors:

termination, execution, problem_selection.

The Active_Strategy model has these submodels :
Active_Problem, Active_Goal.

The Active_Strategy model has fields that references these models :
Strategy, Active_Strategy, Active_Problem, Active_Goal, Plan.

These models have a field that is a Active_Strategy model :
Plan, Active_Strategy, Active_Problem, Active_Goal,
BEFORE_AND_AFTER, SEQUENTIAL_ALTERNATES,
SEQUENTIAL_ALTERNATES_KEEP_BEST.

The key field for this model is strategy

This model may be extended with user-defined fields.

strategy -- a Strategy field of model Active_Strategy
The Strategy:..
Default: None -- this is a key field

super -- a Active_Strategy field of model Active_Strategy
The strategy this is a sub of
Properties: java_method=purcm Export-Only Field

remark -- a String field of model Active_Strategy
Remarks about the performance of the strategy' on the owner Plan.
Default: none

date_activated -- a Date field of model Active_Strategy
The Date at which this Active_Strategy was activated for this Plan. Note that some Problem_Sets of the Strategy may specify that only Problems changed after the Strategy was activated should be addressed. (See the last_change_after_activated field of Problem_Set.) In that case, only Problems changed after this date_activated would be addressed. This allows the user to make some changes and then work only the effects of those changes.

Properties: Export-Only Field

reset_date_activated -- a Date field of model Active_Strategy

Resets the activation date of the strategy to be the current date. This function incurs a 2 second delay. The delay is necessary so that there won't be any problems modified at the same time as the activation date is reset.

Properties: command=True Export-Only Field

run -- a Void field of model Active_Strategy
Run this Active_Strategy, starting with the first sub_strategy. Try to resolve the specified Problems, according to the specified conditions.

Properties: command=True Export-Only Field

stop -- a Void field of model Active_Strategy
Stop this Active_Strategy. Only has an effect if it is currently running.

Properties: command=True; exec_priority=IMMEDIATE Export-Only Field

continue -- a Void field of model Active_Strategy

Continue this Active_Strategy from where it was last stop'd.

Properties: command=True; java_method=continue_running Export-Only Field

running -- a Logical field of model Active_Strategy

If "true", this strategy is currently running on the owner Plan. Setting this to "true" is equivalent to calling 'run'; setting it to "false" returns it to its initial state.

Default: false

Properties: Export-Only Field

stopped -- a Logical field of model Active_Strategy

If "true", this strategy is running but has been stopped (via stop). Setting this to "true" is equivalent to calling 'stop'; setting it to "false" is equivalent to calling 'continue'. If this strategy is not running, then this is necessarily "false", and setting it has no effect.

Default: false

Properties: Export-Only Field

<i>Models</i>	<i>Active_Strategy Model</i>
---------------	------------------------------

run_time - - *a Time field of model Active_Strategy*
The total run Time that has elapsed. This is compared to strategy.max_run_time.
Properties: Export-Only Field

stable_time - - *a Time field of model Active_Strategy*
The total Time that has elapsed since the last improvement was made to the Plan. This is compared to strategy.max_stable_time.
Properties: Export-Only Field

target_achieved - - *a Logical field of model Active_Strategy*
Returns "true" if the target as specified by the strategy.goal and related fields has been achieved. Note that the ultimate goal may still not be achieved, though the target has been.
Properties: Export-Only Field

interaction - - *a Number field of model Active_Strategy*
Returns the sum of interaction values from all active problems
Properties: Export-Only Field

annealing_goodness - - *a Number field of model Active_Strategy*
Returns a number representing the comparative goodness of the plan. Initially the annealing_goodness will return 1. Smaller goodness indicates a better plan, judged by its goals.
Properties: Export-Only Field

resolve_count - - *a Number field of model Active_Strategy*
The number of resolves that have been called by the strategy so far.
Properties: Export-Only Field

permanent - - *a Logical field of model Active_Strategy*
If "true", then this strategy remains active on the Plan even when not running. The Problem Lists and performance information is maintained for every change to the Plan. If "false", then this Active_Strategy will vanish once it completes running. Thus, the only way to see a non-permanent Active_Strategy is to run a Strategy on this Plan and then observe it while it is still running.
Default: true

auto_run - - *a Logical field of model Active_Strategy*
If "true", the Strategy is run after each change to the owner Plan. If "false", this Active_Strategy is kept up-to-date, but not run on each change.

<i>Models</i>	<i>Active_Strategy Model</i>
---------------	------------------------------

There can be only one Active_Strategy for any Plan that has auto_run set to "true". Setting this to "true" will set all other Active_Strategy's of this Plan to "false".
Default: false

background_run - - *a Logical field of model Active_Strategy*
If "true", the Strategy is run as a background activity whenever there has been no planning activity. In this way you can have Strategy's that churn all day trying to find small improvements to the Plan.

Note that when planning activity resumes, the background Strategy will need to stop running first. That may cause a noticeable delay on that first access.

There should be at most one Active_Strategy for any Plan that has background_run set to "true". Setting this to "true" will set all other Active_Strategy's of this Plan to "false".
Default: false

active_problems - - *a list of Active_Problem submodels of model Active_Strategy*
The Problems in the owner Plan that this Active_Strategy is working to resolve and the focus that it is giving to each, as defined by one of the Problem_Sets in the Strategy. Note that if a Problem is selected by more than one Problem_Set, the highest focus value is used.

problems, problems (Date_Range) , problems (Problem_Category) , problems (Date_Range, Problem_Category) - - a List[Problem]field of model Active_Strategy
The Problems detected with this Plan. If passed a Date_Range, only the Problems whose dates overlap are returned. If passed a Problem_Category, only the Problems with that category are returned.

Note that you can pass in one of the special Problem_Category's to get Problems in larger groups. For example, the OPERATION Problem_Category will give all Problems in Operation-related Problem categories. Similarly for RESOURCE, BUFFER, and even ALL.

Properties: Export-Only Field

problem_categories - - *a List[Problem_Category]field of model Active_Strategy*
For each different category of Problem in problems, a Problem_Category is added to this list. It gives you the name of the category (in various forms) plus a list of just the Problems of that category. These sublists are often much easier to deal with than the full problems list.

Properties: Export-Only Field

active_goals -- a list of Active_Goal submodels of model Active_Strategy

The goals of this Active_Strategy and focus for each as defined by the 'strategy'. In addition, the current value(s) of this Plan for each goal.

termination -- an extension selector of model Active_Strategy

Defines the Active_Strategy's progress towards termination, as defined by the 'strategy'. For example, it may be specify to stop after N seconds, to stop after N seconds of no Plan improvements, to stop when the next improved Plan is found, to stop when a certain Cost goal is achieved, etcetera.

Properties: Export-Only Field

Extensions:

NO_PROBLEMS, TARGET_ACHIEVED, RESOLVE_COUNT_EXCEEDED, MANUAL.

execution -- an extension selector of model Active_Strategy

execution specifies when and where the sub strategies are called

Properties: Export-Only Field

Extensions:

SEQUENCE_RUN_ONCE, SEQUENCE_RUN_MULTIPLE, BEFORE_AND_AFTER_SEQUENTIAL_ALTERNATES, SEQUENTIAL_ALTERNATES, KEEP_BEST, PROPORTIONAL_RESOLVES, ORDERED_RESOLVES.

problem_selection -- an extension selector of model Active_Strategy

The problem_selection extension specifies how this strategy will select problems to solve.

Properties: Export-Only Field

Extensions:

PROPORTIONAL_INTERACTION, EARLIEST_PROBLEM_START, SORT_BY_EXPRESSION.

owner -- a Plan field of model Active_Strategy

Properties: Export-Only Field

5.1.3.4.1 Active_Problem Model

Active_Problem -- a submodel of model Active_Strategy

The Problems in the 'owner' Plan that this Active_Strategy is working to resolve and the Focus that it is giving to each, as defined by one of the Problem_Sets in the 'strategy'. Note that if a Problem is selected by more than one Problem_Set, the highest Focus value is used.

The Active_Problem model has fields that references these models :
Problem, Active_Strategy.

These models have a field that is a Active_Problem model :
Active_Strategy.

The key field for this model is problem

problem -- a Problem field of model Active_Problem

A Problem to be addressed by this Active_Strategy with Focus:

Default: None -- this is a key field

Properties: Export-Only Field

focus -- a Percentage field of model Active_Problem

A Percentage that indicates how much focus will be given to the 'problem' by this Active_Strategy. The greater the Percentage, the more attention that the 'problem' will get.

This is computed from the Strategy's 'problem_seis' and 'feasible_focus'.

Properties: Export-Only Field

unresolvable -- a Logical field of model Active_Problem

Set to true when the strategy detects that this problem is not resolvable using the current change categories. Site edits, and in some cases changes to the plan, may cause the problem to become resolvable again (but in many cases this field will not be updated for some time).

Default: false

Properties: Export-Only Field

interaction -- a Number field of model Active_Problem

A number indicating the impact on annealing goodness. It's value depends on problem size and Active_Strategy's goals.

Model	Active_Problem Model
-------	----------------------

Properties: Export-Only Field

must_resolve - - *a Logical field of model Active_Problem*
 Indicates whether this is a must_resolve problem. This field is true if 'problem' is a member of at least one Problem_Set whose must_resolve is true.
 Default: false

Properties: Export-Only Field

owner - - *a Active_Strategy field of model Active_Problem*

Properties: Export-Only Field

Model	Active_Goal Model
-------	-------------------

5.1.3.4.2 Active_Goal Model

Active_Goal - - *a submodel of model Active_Strategy*

The goals of this Active_Strategy and focus for each as defined by the 'strategy'. In addition, the current value(s) of this Plan for each goal.

The model has selectors:
 goal.

The Active_Goal model has fields that references these models :
Strategy_Goal, Active_Strategy.

These models have a field that is a Active_Goal model :
Active_Strategy.

The key field for this model is strategy_goal

strategy_goal - - *a Strategy_Goal field of model Active_Goal*
 The goal defined in the Strategy for which this models the current value for this Plan.
 Default: None - - this is a key field

Properties: Export-Only Field

goal - - *on extension selector of model Active_Goal*
 This is defined by strategy_goal.goal'. This extension will add fields that contain the absolute values for the current value for this Plan, the values before adjustment to a comparable numeric value.
 Properties: Export-Only Field

Extensions:
FEASIBILITY, MINIMIZE_PROBLEM_COUNT, MINIMIZE_PROBLEMS, MINIMIZE_LATENESS, WEIGHTED_LATENESS, WEIGHTED_SHORTNESS, MINIMIZE_COST, MAXIMIZE_PROFIT, MAXIMIZE_REVENUE.

adjusted_value - - *a Number field of model Active_Goal*
 The adjusted numeric value of this 'goal' for the Plan, comparable to the 'adjusted_target' for this goal and the 'adjusted_value's of other goals.
 Properties: Export-Only Field

Models	Active_Goal_Model
--------	-------------------

adjusted_target -- *a Number field of model Active_Goal*
The target value of this goal', after adjustments. There will often be an extension-specific target field that is before adjustments.

Achieving values higher than the target may receive much less focus (focus_beyond_target) than achieving the target (focus_to_target).

Properties: Export-Only Field

focus_value -- *a Number field of model Active_Goal*

This is the adjusted numeric value weighted by the target_focus' and 'over_target_focus' as it will be combined with the other goals. The sum of the 'focus_value's for all the goals is the overall "global goodness" value.

If 'adjusted_value' is less than 'adjusted_target', this value is simply 'focus_to_target * adjusted_value'. If 'adjusted_value' is over 'adjusted_target', then this value is 'focus_beyond_target * (adjusted_value - adjusted_target) + focus_to_target * adjusted_target'.

Properties: Export-Only Field

owner -- *a Active_Strategy field of model Active_Goal*

Properties: Export-Only Field

Models	Problem_Category_Model
--------	------------------------

5.2 Problem_Category_Model

Problem_Category -- *an independent (top-level) model*

A Problem_Category models a category of Problems. It is not a Problem, nor does it contain Problems. Rather, it tells you about a kind of Problem. It is a lot like the Model_Type model, which models a type of model.

The Problem_Category model provides the name, the short_name, the long_name, and a description of the nature of Problems of that category. Further, it provides a list of the Problem_Field's that are specific to that category.

Each planning model that provides a List(Problem) also provides a List(Problem_Category). Those Problem_Category's can be passed to the field to get just the Problems in that Problem_Category. These sublists are often much easier to deal with than the full list of Problems. In particular, coupled with the list of the fields in this Problem_Category, formulating informative Problem Reports is much easier.

There are several special Problem_Category's that are not really Problem_category' extensions, but rather a set of Problem_category's. For example, the Problem_Category_OPERATION includes all Problem_category's that relate to Operations. Similarly for RESOURCE, BUFFER, and ALL. ALL includes all Problem_category's.

The key field for this model is name

This model may be extended with user-defined fields.

name -- *a Symbol field of model Problem_Category*
This is the name used as the 'category' extension selector value in the Problem model.
Default: None -- this is a key field

Properties: Export-Only Field

short_name -- *a Symbol field of model Problem_Category*
The short name for the Problem_category' extension.

Properties: Export-Only Field

full_name -- *a Symbol field of model Problem_Category*
The full name for the Problem_category' extension.

Properties: Export-Only Field

Model	Problem_Category Model
-------	------------------------

```

description -- a String field of model Problem_Category
A description of the Problem_Category' extension.
Properties:      Export-Only Field

remark -- a String field of model Problem_Category
Additional user-provided discussion about this Problem_Category' extension.
Default:      none

super_categories -- a List(Problem_Category) field of model Problem_Category
The direct super_categories of this Problem_Category.
Properties:      Export-Only Field

sub_categories -- a List(Problem_Category) field of model Problem_Category
The direct sub_categories of this Problem_Category.
Properties:      Export-Only Field

leaf_categories -- a List(Problem_Category) field of model Problem_Category
The leaf sub_categories of this Problem_Category. A leaf Problem_Category is one
that contains no other Problem_Category's; one that identifies only a single 'category'
extension of Problem.
Properties:      Export-Only Field

fields -- a List(Symbol) field of model Problem_Category
The Fields provided by this Problem_Category' extension.
Properties:      Export-Only Field

```

Model	Horizon Model
-------	---------------

5.3 Horizon Model

Horizon -- an independent (top-level) model

Horizon models how to break up time into buckets. For example, forecasting is often done in monthly buckets, but is not uncommonly done in quarters, weeks, and other variations. Further, there will often be smaller buckets in the near-term followed by larger buckets. For example, weeks for the first 3 months, months for the next 6 months, and then quarters for the rest of the horizon.

Most often the same bucketing is desired in most places. Inconsistent buckets can prevent meaningful comparisons and analyses. However, there is a very real need to be able to choose different bucketing in different areas. Thus, rather than have to define and maintain the same bucket specification in each of hundreds of models, the Horizon model allows you to define named specifications that can then be reused by name in all of those models that should use a common bucket specification.

The model has selectors:
bucket_spec.

These models have a field that is a Horizon model :
Seller, Horizon_Bucket_Start, HORIZON, SHARED_USE,
BUCKETED_NESTED_SORT.

The key field for this model is name
This model may be extended with user-defined fields.

name -- a Symbol field of model Horizon
The name of this Horizon.
Default: None -- this is a key field

description -- a String field of model Horizon
A description of this Horizon.
Default: none

bucket_spec - - an extension selector of model Horizon

The bucket_spec extension and related fields define how to compute the time buckets that the planning horizon or Date_Range should be broken into. It always snaps the ending date to the end of the last bucket, even if the plan horizon ends in the middle of the bucket. Because the last bucket will always be consistent in size with the other buckets, the user will be able to see more easily how changes to the plan horizon affect the buckets. This avoids potential errors which may arise from edits to plan horizon which move the horizon end less than one bucket size.

Default: ONE

Extensions:

ONE, MONTHS, WEEKS, DAYS, DATES,

buckets (Date_Range) - - a List(Date_Range) field of model Horizon

This will return list of buckets defined by currently selected 'bucket_spec' extension.

If no argument is passed, this returns list of buckets within planning horizon(plan.start to plan.end). Start date for the first bucket will be same as the plan.start and the end date for the last bucket in the list will be same as plan.end. !!!!UNIMPLEMENTED!!!!

If Date_Range argument is given, this will return list of buckets within that

Date_Range. First bucket's start will be same as the start date of the Date_Range passed in as argument. Similarly, the end date of last bucket will be same as the end date of the Date_Range passed in as argument.

Properties: Export-Only Field

5.3.1 bucket_spec submodels of model Horizon
5.3.2 Horizon_Bucket_Start Model

Horizon_Bucket_Start - - a submodel of model Horizon

List of bucket_starts specifying bucket boundaries.

The Horizon_Bucket_Start model has fields that references these models :
Horizon.

These models have a field that is a Horizon_Bucket_Start model :
DATES.

The key field for this model is start

start - - a Date field of model Horizon_Bucket_Start
start date of the bucket.
Default: None -- this is a key field

owner - - a Horizon field of model Horizon_Bucket_Start

Properties: Export-Only Field

Model	Strategy Model
-------	----------------

5.4 Strategy Model

Strategy -- *an independent (top-level) model*

An enumeration value used by the Strategy_Change_category' field. See that field for details.

The Strategy allows a user to drive the automated planning algorithms. There are five basic things that can be specified by a Strategy: the problems to focus on, the changes to the plan to focus on, the goal trying to be achieved (what is a good plan, what is better), the termination condition (when is this Strategy "done"), and sub-strategies which allows a planning flow to be assembled.

The problems to focus on are specified by the Problem_Set submodels. Each Problem_Set defines a category of Problems, possibly some filtering expressions or parameters, and a focus level (which is relative to any other Problem_Sets). For example, you can specify that the Strategy focuses heavily on BUFFER Problems, lightly on RESOURCE Problems, and not at all on PROMISE_PLAN Problems. Or in more detail, heavily on NEGATIVE_ON_HAND Buffer Problems and moderately on LOW_ON_HAND Buffer Problems.

The plan changes to focus on making are specified by the Strategy_Change submodels. Strategy_Change submodels define the relative 'focus' that should be given to each category of problem resolutions. For example, to avoid making anything later one can specify that MOVE_OUT is disallowed.

The goals of this Strategy -- the computation of the quality of a plan -- are specified by the Strategy_Goal submodels. Each strategy goal defines both a goal (e.g. MAXIMIZE_REVENUE) and the relative weight of this goal vs. the other goals of the strategy.

When this Strategy is "done" is specified by the 'termination' extension and related fields. For example, one might specify that the strategy is done when a feasible plan is found in which lateness is below a given bound.

Finally, the Sub_Strategy submodels specify Strategies that are run as part of this Strategy.

The model has selectors: termination, execution, problem_selection,

The Strategy model has these submodels :

Model	Strategy Model
-------	----------------

Problem_Set, Strategy_Change, Strategy_Lock, Strategy_Goal.

The Strategy model has fields that references these models :

Problem_Set, Strategy_Change, Strategy_Lock, Strategy_Goal.

These models have a field that is a Strategy model :

Active_Strategy, Problem_Set, Strategy_Change, Strategy_Lock, Strategy_Goal, Ordered_Sub_Strategy, BEFORE_AND_AFTER, SEQUENTIAL_ALTERNATES, SEQUENTIAL_ALTERNATES_KEEP_BEST, Ranked_Sub_Strategy.

The key field for this model is name

name -- *a Symbol field of model Strategy*

The name of this Strategy.

Default: None -- this is a key field

description -- *a String field of model Strategy*

A description of the goals, approach, usage, and usefulness of this Strategy.

Default: no empty String

deterministic_resolvers -- *a Logical field of model Strategy*

Default FALSE. When TRUE causes the strategy to always set heat=0. This should

cause most resolvers to behave deterministically.

Default: false

deterministic_problem_selection -- *a Logical field of model Strategy*

Default FALSE. When TRUE causes the strategy to always select the problem for which the product of the interaction and focus value is the largest.

Default: false

problem_sets -- *a list of Problem_Set submodels of model Strategy*

Each Problem_Set defines a set of Problems to be addressed by this Strategy. The Problems are specified by category and tolerances within a certain horizon. The relative 'focus' that the Strategy should place on each Problem is also specified.

changes -- *a list of Strategy_Change submodels of model Strategy*

Defines the relative 'focus' that should be given to each Change_Category that can be performed to resolve Problems. For example, to avoid making anything later, you can specify that MOVE_OUT is disallowed (0% focus).

Models	Strategy Model
--------	----------------

Change_Category's that do not appear in this List get the 'default_change_focus'.

locks - - - *a list of Strategy_Lock submodels of model Strategy*

A list of specifications that describe what objects are to remain locked/unchanged for the duration of the strategy execution. For example, users can lock Operation_Plan's using Strategy_Lock. The effects of multiple Strategy_Locks are additive; should any Strategy_Lock consider an Operation_Plan 'locked', then it will be treated as such.

default_change_focus - - - *a Percentage field of model Strategy*

The 'focus' for Change_Category's not explicitly defined in 'changes'. By setting this to 0%, no change will be tried unless it is mentioned in 'changes'. By setting this greater than 0%, all changes will be tried unless explicitly given 'focus' of 0% in 'changes'.
Default: 100%

goals - - - *a list of Strategy_Goal submodels of model Strategy*

The goals of this strategy. This defines global goodness.

termination - - - *an extension selector of model Strategy*

Defines how the Strategy decides to terminate a run. For example, it may be specify to stop after N seconds, to stop after N seconds of no Plan improvements, to stop when the next improved Plan is found, to stop when a certain Cost goal is achieved, etcetera.

Note that by default, the Strategy will run forever!

Default: NO_PROBLEMS

Extensions:

NO_PROBLEMS, TARGET_ACHIEVED, RESOLVE_COUNT_EXCEEDED, MANUAL.

execution - - - *an extension selector of model Strategy*

The execution extension allows users to specify Sub_Strategy's that will be performed as part of this Strategy. It will also define how these Sub_Strategy's will be executed.

For example, SEQUENCE_RUN_ONCE execution extension specifies

Sub_Strategy's that will run to termination in a particular sequence exactly once. PROPORTIONAL_RESOLVE execution extension will repeatedly select one of the Sub_Strategy probabilistically and resolve one of it's Active_Problem.

A Strategy "run" continues until one of it's termination conditions are met. A "run" consists of a series of "searches". A "search" is a series of Problem "resolves" that result in a new plan. If a search results in a better Plan, where better is defined by the Strategy 'goals', then the search is termed a "success".

Default: SEQUENCE_RUN_ONCE

Models	Strategy Model
--------	----------------

Extensions:

SEQUENCE_RUN_ONCE, SEQUENCE_RUN_MULTIPLE, BEFORE_AND_AFTER_SEQUENTIAL_ALTERNATES, SEQUENTIAL_ALTERNATES, KEEP_BEST, PROPORTIONAL_RESOLVES, ORDERED_RESOLVES,

problem_selection - - - *an extension selector of model Strategy*

The problem_selection extension specifies how this strategy will select problems to solve.
Default: PROPORTIONAL_INTERACTION

Extensions:

PROPORTIONAL_INTERACTION, EARLIEST_PROBLEM_START, SORT_BY_EXPRESSION.

max_stable_time - - - *a Time field of model Strategy*

This Strategy will terminate if it does not find an improvement to the plan within 'max_stable_time'.
Default: infinite

max_stable_resolve_count - - - *a Number field of model Strategy*

This Strategy will advance to the next annealing run or terminates if it could not find an improvement to the plan within 'max_stable_resolve_count' problem resolutions.

Default: infinite

Properties: obsolete=True

max_run_time - - - *a Time field of model Strategy*

This Strategy will terminate if it runs for longer than 'max_run_time'.
Default: infinite

max_resolve_count - - - *a Number field of model Strategy*

This Strategy will terminate if it attempts more than 'max_resolve_count' resolves.
Default: infinite

max_heat - - - *a Quantity field of model Strategy*

Strategy will start every annealing run with this value of maximum heat.
Default: 100

min_heat - - - *a Quantity field of model Strategy*

Lowest annealing heat.
Default: 0.0

Model	Strategy Model
-------	----------------

```

run (Plan) -- a Void field of model Strategy
Run the Strategy. Try to resolve the specified Problems, according to the specified
conditions.
Properties:    command=True Export-Only Field

do_before -- a Expression field of model Strategy
An Expression that is passed an Active_Strategy as 'w' and is executed before the
Strategy run.
Default: NONE

do_after -- a Expression field of model Strategy
An Expression that is passed an Active_Strategy as 'w' and is executed after the Strat-
egy run.
Default: NONE

do_before_resolve -- a Expression field of model Strategy
An Expression that is passed an Active_Problem as 'w' and is executed before the
problem resolution.
Default: NONE

```

Models	Problem_Set Model
--------	-------------------

5.4.1 Problem_Set Model

Problem_Set -- a submodel of model Strategy

Each *Problem_Set* defines a set of Problems to be addressed by this Strategy. The Problems are specified by category and tolerances within a certain horizon. The relative 'Focus' that the Strategy should place on each Problem is also specified.

The model has selectors:
category.

The *Problem_Set* model has fields that references these models :
Strategy.

These models have a field that is a *Problem_Set* model :
Strategy.

The key field for this model is category
This model may be extended with user-defined fields.

```

fence -- a Horizon_Date field of model Problem_Set
This Problem_Set will only identify a Problem with 'dates.start' prior to this 'fence'.
This 'fence' is computed as a Horizon_Date relative to 'plan.current'.
Default: oo_future

```

```

min_time -- a Time field of model Problem_Set
This Problem_Set will only identify a Problem with 'dates.time' >= this 'min_time'.
The 'dates' in the problem do not necessarily reflect the duration over which prob-
lem exists. In some cases, the dates associated with the problem are the start and end
dates of the flow plan which has the problem. The 'min_time' field cannot be used to
filter problems based on the time between the problem occurring and the end of the
planning horizon -- unless, of course, the source of the problem's dates happens to
accidentally overlap that region. The meaning of 'min_time' varies with problem cate-
gory.
Default: 0

last_change_after_activated -- a Logical field of model Problem_Set
If 'true', then this Problem_Set will only identify Problems with 'last_change' after
'date_activated' of the Active_Strategy.

```


*Models**Problem_Set Model*

This allows the user to make some changes and then have a Strategy work on only the effects of those changes. Given a Strategy that has Problem_Sets with `last_change_after_activated` set 'true', you can activate it on a Plan, make changes to that Plan, and then run that Active_Strategy. Only the Problems created or changed due to the changes made to that Plan since you activated the Strategy will be addressed.

Default: false

category - - *an extension selector of model Problem_Set*

A Problem category that this Strategy will address. This extension adds similar fields as the Problem category adds. Here they specify tolerances on what is allowable.

Problems within the tolerance are not addressed by this Strategy.

There are also additional category extensions that are generalizations of the Problem category extensions. For example, RESOURCE will match any Resource, Plan related Problem, including OVERLOAD, UNDERLOAD, OVERTIME, and the rest.

Default: None - - this is a key field

Extensions:

REQUEST_NOT_PLANNED, REQUEST_PLANNED_LATE,
REQUEST_PLANNED_EARLY, REQUEST_PLANNED_SHORT,
REQUEST_PLANNED_EXCESS, PROMISE_NOT_PLANNED,
PROMISE_PLANNED_LATE, PROMISE_PLANNED_EARLY,
PROMISE_PLANNED_SHORT, PROMISE_PLANNED_EXCESS,
ACCEPTANCE_NOT_PLANNED, ACCEPTANCE_PLANNED_LATE,
ACCEPTANCE_PLANNED_EARLY, ACCEPTANCE_PLANNED_SHORT,
ACCEPTANCE_PLANNED_EXCESS, PLANNED_BEFORE_CURRENT,
UNRELEASED_NEEDS_RELEASE, INCONSISTENT_OPLAN, OPERATION,
REQUEST_PROMISED_LATE, REQUEST_PROMISED_EARLY,
REQUEST_PROMISED_SHORT, REQUEST_PROMISED_EXCESS,
PROMISE_NOT_OFFERED, PROMISE_NOT_ACCEPTED,
ACCEPTANCE_INCONSISTENT, REQUEST_QUEUED, REQUEST,
REQUEST_PLAN_PROMISE, PROMISE_PLAN, REQUEST_PROMISE,
DELIVERY_REQUEST_NOT_COORDINATED,
DELIVERY_PROMISE_NOT_COORDINATED,
DELIVERY_ACCEPTANCE_NOT_COORDINATED,
SUPPLY_PLANNED_LATE, SUPPLY_PLANNED_EARLY,
SUPPLY_PLANNED_SHORT, SUPPLY_PLANNED_EXCESS,
SUPPLY_PLANNED_LATE, SUPPLY_PROMISED_EARLY,
SUPPLY_PROMISED_SHORT, SUPPLY_PROMISED_EXCESS, SUPPLY,
SUPPLY_PLAN, SUPPLY_PROMISE, UNIDENTIFIED_OP_STATE, UNCONSOLIDATED, UNCOORDINATED, CONSOLIDATION, OVERSIZE,
CONSOLIDATION_UNDERSIZE, OVERLOAD, OVERSIZE,

*Models**Problem_Set Model*

BUCKET OVERSIZE, UNDERLOAD, RESOURCE, NEGATIVE_ON_HAND,
OVER_FLOW_LIMIT, NEGATIVE_ON_HAND_AT_END,
LOT_OVER_CONSUMED, LOT_NOT_CONSUMED,
LOT_NOT_PRODUCED, LOT_OVER_PRODUCED, LOW_ON_HAND,
EXCESS_ON_HAND, EXCESS_ON_HAND_AT_END, BUFFER.

min_cost - - *a Money field of model Problem_Set*

This Strategy only addresses Problems with cost >= this min_cost.

Default: 0

must_resolve - - *a Logical field of model Problem_Set*

Specifies whether problems of this problem set must be resolved. Problem sets for which this field is false are desirable to resolve but not necessary to resolve. SDP behavior is greatly altered by this field.

Default: false

focus - - *a Percentage field of model Problem_Set*

A Percentage that indicates how much focus should be given to Problems identified by this Problem_Set, relative to the other Problem_Sets in the owner Strategy. The greater the Percentage, the more attention that those Problems will get. If a single Problem is identified by more than one Problem_Set, the higher focus is used.

Default: 100%

feasible_focus - - *a Percentage field of model Problem_Set*

Weights the focus of 'feasible' Problems relative to Problems that are not 'feasible'. If feasible, the Problem will receive focus equal to focus multiplied by this value. If "0%", then 'feasible' Problems are not addressed at all. If "100%", then 'feasible' Problems are addressed as often as infeasible ones.

Default: 100%

horizon - - *a Horizon_Date field of model Problem_Set*

Obsolete! This field has been renamed 'tence', to prevent confusion with the new Horizon model. Please use 'tence' instead. horizon will be removed in a later release.

Default: oo_future

owner - - *a Strategy field of model Problem_Set*

Properties: Export-Only Field

5.4.2 Strategy_Change Model

Strategy_Change -- a submodel of model Strategy

Defines the relative 'focus' that should be given to each Change_Category that can be performed to resolve Problems. For example, to avoid making anything later, you can specify that MOVE_OUT is disallowed (0% focus).

Change_Category's that do not appear in this List get the 'default_change_focus'.

The model has selectors:

change_category.

The Strategy_Change model has fields that references these models :

Strategy.

These models have a field that is a Strategy_Change model :

Strategy.

The key field for this model is change_category

change_category - - *an extension selector of model Strategy_Change*
The Change_Category that should be tried with 'focus' relative weighting. Some examples:

UPSTREAM (resolve a Problem by adjusting an upstream Operation_Plan),
DOWNSTREAM (resolve a Problem by adjusting a downstream Operation_Plan),
MOVE_IN (move an Operation_Plan to an earlier Date),
MOVE_OUT (move an Operation_Plan to a later Date),
RESIZE_MORE (increase the Quantity of an Operation_Plan),
RESIZE_LESS (decrease the Quantity of an Operation_Plan),
USE_ALTERNATE_OPERATION (use alternate Operations), or
USE_ALTERNATE_RESOURCE (use alternate Resources).

By using different combinations of these, you can control what the Strategy will generally do to the plan. For example, by setting MOVE_IN and/or RESIZE_MORE to 100% focus and MOVE_OUT and RESIZE_LESS to 0%, you will tend to increase WIP, but will not increase lateness or shortness. By setting DOWNSTREAM to 0%, you will tend to move Buffer Problems upstream to the raw material Buffers.

For another example, suppose for a given strategy, a MOVE_IN of 75 and a MOVE_OUT of 25 are defined. These focus numbers stay the same, but they are used by resolvers along with the current state of the plan. For instance, the resource problem resolvers choose whether to move in, out, or off. They prefer moved to open space closest to the current plan. If open space before the start of the current operation plan is two weeks away, the resolvers weight more heavily the move in over the move out.

The RESIZE_LESS and RESIZE_MORE change_categories should be thought of in terms of Operation_Plan quantity, not Flow_Plan quantity. For example, consider a Flow_Plan consuming 10 from a buffer, which would appear as -10 in the editor. A RESIZE_MORE applied to that Flow_Plan would change it to -20 rather than -5.

An additional strategy change_category that can affect buffer plan resolution is USE_ALTERNATE_OPERATION. As with other strategy focus change_categories, the focus can vary. If set to zero, RHYTHM does not use alternate operations.

Although changing to an alternate operation cannot always help solve buffer problems, there are situations where it can. An example is transportation, where moving from ground to air transportation may solve a problem related to receiving materials when needed.

Default: None -- this is a key field

Extensions:

MOVE_IN, MOVE_OUT, SPLIT, USE_MORE, USE_LESS,
USE_ALTERNATE_OPERATION, USE_ALTERNATE_RESOURCE,
USE_EFFECTIVE_ALTERNATE, INCREASE_CAPACITY,
DECREASE_CAPACITY, RESIZE_MORE, RESIZE_LESS, UPSTREAM,
DOWNSTREAM.

focus - - *a Percentage field of model Strategy_Change*

A relative weighting for how often the change_category should be tried while resolving Problems. "0%" indicates that the change_category should never be done.

"100%" indicates that the change_category should be tried twice as often as "50%", but half as often as "200%". default is "100%". If the 'default_change_focus' is 0% and is "0%" otherwise.

Default: 100% if 'default_change_focus' is 0%, 0% otherwise

owner - - *a Strategy field of model Strategy_Change*

Properties: Export-Only Field

5.4.3 Strategy_Lock Model

Strategy_Lock -- a submodel of model Strategy

A list of specifications that describe what objects are to remain locked/unchanged for the duration of the strategy execution. For example, users can lock Operation_Plan using Strategy_Lock. The effects of multiple Strategy_Locks are additive; should any Strategy_Lock consider an Operation_Plan 'locked', then it will be treated as such.

The model has selectors:
spec.

The Strategy_Lock model has fields that references these models :
Strategy.

These models have a field that is a Strategy_Lock model :
Strategy.

The key field for this model is name

name -- a Symbol field of model Strategy_Lock
The name of this Strategy_Lock.
Default: None -- this is a key field

spec -- an extension selector of model Strategy_Lock
The spec extension defines the additional fields required to specify locking criteria. For example, OPERATION_PLAN_RANK_RANGE.spec extension specifies a range which will lock Operation_Plan whose rank lies within it. The OPERATION_PLAN_RANK_EXPRESSION.spec extension allows user to use OIL expressions to determine which Operation_Plan get locked.
Default: OPERATION_PLAN_RANK_RANGE
Extensions:
OPERATION_PLAN_RANK_RANGE,
OPERATION_PLAN_RANK_EXPRESSION.

owner -- a Strategy field of model Strategy_Lock

Properties: Export-Only Field

5.4.4 Strategy_Goal Model

Strategy_Goal -- a submodel of model Strategy

The goals of this strategy. This defines global goodness.

The model has selectors:
goal.

The Strategy_Goal model has fields that references these models :
Strategy.

These models have a field that is a Strategy_Goal model :
Strategy, Active_Goal.

The key field for this model is goal

goal -- an extension selector of model Strategy_Goal
The goal to be given 'focus' emphasis in the comparison of planning alternatives. The extension will add fields that define how to convert the absolute measure for the plan to a generic number that is numerically comparable to other goals of this Strategy.
Default: None -- this is a key field
Extensions:
FEASIBILITY, MINIMIZE_PROBLEM_COUNT, MINIMIZE_PROBLEMS,
MINIMIZE_LATENCY, WEIGHTED_LATENCY,
MINIMIZE_SHORTNESS, WEIGHTED_SHORTNESS, MINIMIZE_COST,
MAXIMIZE_PROFIT, MAXIMIZE_REVENUE.

focus_to_target -- a Percentage field of model Strategy_Goal
The focus (emphasis, weighing) on achieving the target for the 'goal' relative to other goals.
Default: 100%

focus_beyond_target -- a Percentage field of model Strategy_Goal
The focus (emphasis, weighing) on achieving values over the target for the 'goal' relative to other goals.
Default: 0%

owner -- a Strategy field of model Strategy_Goal

Properties: Export-Only Field

Models	execution submodels of model Strategy
--------	---------------------------------------

5.4.5 execution submodels of model Strategy

5.4.6 Ordered_Sub_Strategy Model

Ordered_Sub_Strategy -- a submodel of model Strategy

An Ordered_Sub_Strategy that will be performed in particular sequence as part of the 'owner' Strategy.

The Ordered_Sub_Strategy model has fields that references these models : Strategy.

These models have a field that is a Ordered_Sub_Strategy model :
SEQUENCE_RUN_ONCE, SEQUENCE_RUN_MULTIPLE,
ORDERED_RESOLVES, SEQUENCE_AFTER_EACH_RUN.

The key field for this model is sequence

sequence -- a Number field of model Ordered_Sub_Strategy
The lower the Number, the earlier the Strategy is performed. No two Ordered_Sub_Strategy's of a Strategy may have the same sequence number.
Default: None -- this is a key field

strategy -- a Strategy field of model Ordered_Sub_Strategy
A Strategy to be performed on the subset of Problems selected by the 'owner' Strategy.
Default: [unspecified]

owner -- a Strategy field of model Ordered_Sub_Strategy

Properties: Export-Only Field

5.4.7 Ranked_Sub_Strategy Model

Ranked_Sub_Strategy -- a submodel of model Strategy

A Ranked_Sub_Strategy that can be selected and performed by the 'owner' Strategy.

The Ranked_Sub_Strategy model has fields that references these models : Strategy.

These models have a field that is a Ranked_Sub_Strategy model :
PROPORTIONAL_RESOLVES.

Models	Ranked_Sub_Strategy Model
--------	---------------------------

The key field for this model is strategy

rank -- a Percentage field of model Ranked_Sub_Strategy
A Percentage that indicates how much focus will be given to this Sub_Strategy. The greater the Percentage, the more time will be spent performing this Sub_Strategy.
Default: 100%

strategy -- a Strategy field of model Ranked_Sub_Strategy
A Strategy to be performed on the subset of Problems selected by the 'owner' Strategy.
Default: None -- this is a key field

owner -- a Strategy field of model Ranked_Sub_Strategy

Properties: Export-Only Field

Models	Unit Model
--------	------------

5.5 Unit Model

Unit -- *an independent (top-level) model*

A Unit defines the size (Quantity) of one unit of an Item, Product, or Operation. Effectively, it relates "unit of Item" to a "unit of Measure". For instance, one unit of Paint304 may be "1 gal", "2 kg", and/or "\$4.25".

The Unit model has these submodels :

Unit_Quantity.

The Unit model has fields that references these models :

Operation, Buffer, Product_Root, Item, Product_Group, Unit_Quantity.

The key field for this model is preferred_measure

preferred_measure -- *a Measure field of model Unit*

The preferred Measure to be used when displaying Quantities of this Unit. This Measure must be specified in the quantities, or be the "unitless" Measure which is simply the number of units.

For example, if one unit of Paint304 is "1 gal" and "2 kg", then this could be volume, mass, or unitless; giving "10 gal", "20 kg", or "10", respectively, for ten units of Paint304.

Default: None -- this is a key field

discrete -- *a Logical field of model Unit*

If "true", then this (the one that is measured) can only exist in whole units. In that case, the Quantity used will always be rounded up to a whole number of units of this. Note that a whole unit of this may not be a whole unit of Measure.

Default: false

quantities -- *a list of Unit_Quantity submodels of model Unit*

Specifies zero or more measurements of this Unit. For example, this Unit may be "10kg" in mass. This Unit may also be "16 cm" in height, "4 liters" in volume, and "\$120" in money. Each of those four would be specified as a separate Unit_Quantity. The default Measure used for this Unit is specified in the preferred_measure field.

Models	Unit Model
--------	------------

Note that all Units may be expressed in the "unitless" Measure, which is simply the number of those Units.

convert (Measure_Unit, Measure) -- *a Quantity field of model Unit*

Returns the Quantity of this Unit with the given unit of Measure that is equivalent to the given Quantity. Both Measures must be defined in quantities, or be "unitless" meaning that number of this Unit. If "discrete" is "true", then this will round up to a whole number of these Units (which is not necessarily a whole number of the requested unit of Measure).

Properties: Export-Only Field

5.5.1 Unit_Quantity Model

Unit_Quantity -- *a submodel of model Unit*

Specifies zero or more measurements of this Unit. For example, this Unit may be "10kg" in mass. This Unit may also be "16 cm" in height, "4 liters" in volume, and "\$120" in money. Each of those four would be specified as a separate Unit_Quantity. The default Measure used for this Unit is specified in the preferred_measure field.

Note that all Units may be expressed in the "unitless" Measure, which is simply the number of those Units.

The Unit_Quantity model has fields that references these models :

Unit.

These models have a field that is a Unit_Quantity model :

Unit.

The key field for this model is quantity

quantity -- *a Quantity field of model Unit_Quantity*

Specifies a measurement of this Unit. This is the Quantity of that Measure that is equivalent to this Unit. Each Unit_Quantity must have a unique Measure (e.g., two different measurements of a Unit cannot have volume Measure).

Default: None -- this is a key field

owner -- *a Unit field of model Unit_Quantity*

Properties: Export-Only Field

5.6 Profile_Number Model

Profile_Number -- *an independent (top-level) model*

A List(Profile_Number) is used to profile a Number over time. A Profile_Number defines for a Number a new 'value' at 'date' and new 'rate' continuing from 'date' until the next Profile_Number. For example, at "95-07-23" the Number may change to "1200" and the rate may change to "50/hr". It will continue at "50/hr" until the date of the next Profile_Number.

Note that the new 'value' may be a discrete jump at 'date', independent of the 'rate's before and after that 'date'. Thus, a Profile_Number may correspond to a discrete jump, but the rate may remain the same. Similarly, a Profile_Number may correspond to a 'rate' change on a 'date' where there is no discrete jump in 'value'.

The key field for this model is date

date -- *a Date field of model Profile_Number*

The Date at which this change in either 'value' or 'rate' occurs.

Default: None -- this is a key field

Properties: Export-Only Field

value -- *a Number field of model Profile_Number*

The new Number value at 'date'.

Properties: Export-Only Field

rate -- *a Quantity field of model Profile_Number*

The new Number rate of change (per unit Time) which continues until the next Profile_Number.

Properties: Export-Only Field

5.7 Profile_Percentage Model

Profile_Percentage -- an independent (top-level) model

A List(Profile_Percentage) is used to profile a Percentage over time. A Profile_Percentage defines for a Percentage a new 'value' at 'date' and new 'rate' continuing from 'date' until the next Profile_Percentage. For example, at '95-07-23' the Percentage may change to "88%" and the rate may change to "2%/hr". It will continue at "2%/hr" until the 'date' of the next Profile_Percentage.

Note that the new 'value' may be a discrete jump at 'date', independent of the 'rate's before and after that 'date'. Thus, a Profile_Percentage may correspond to a discrete jump, but the rate may remain the same. Similarly, a Profile_Percentage may correspond to a 'rate' change on a 'date' where there is no discrete jump in 'value'.

The key field for this model is date

date -- a Date field of model Profile_Percentage

The Date at which this change in either 'value' or 'rate' occurs.

Default: None -- this is a key field

Properties: Export-Only Field

value -- a Percentage field of model Profile_Percentage

The new Percentage value at 'date'.

Properties: Export-Only Field

rate -- a Quantity field of model Profile_Percentage

The new Percentage rate of change (per unit Time) which continues until the next Profile_Percentage.

Properties: Export-Only Field

5.8 Profile_Quantity Model

Profile_Quantity -- an independent (top-level) model

A List(Profile_Quantity) is used to profile a Quantity over time. A Profile_Quantity defines for a Quantity a new 'value' at 'date' and new 'rate' continuing from 'date' until the next Profile_Quantity. For example, at '95-07-23' the Quantity may change to "1200 kg" and the rate may change to "50kg/hr". It will continue at "50kg/hr" until the 'date' of the next Profile_Quantity.

Note that the new 'value' may be a discrete jump at 'date', independent of the 'rate's before and after that 'date'. Thus, a Profile_Quantity may correspond to a discrete jump, but the rate may remain the same. Similarly, a Profile_Quantity may correspond to a 'rate' change on a 'date' where there is no discrete jump in 'value'.

The key field for this model is date

date -- a Date field of model Profile_Quantity

The Date at which this change in either 'value' or 'rate' occurs.

Default: None -- this is a key field

Properties: Export-Only Field

value -- a Quantity field of model Profile_Quantity

The new Quantity value at 'date'.

Properties: Export-Only Field

rate -- a Quantity field of model Profile_Quantity

The new Quantity rate of change (in /s units) which continues until the next Profile_Quantity.

Properties: Export-Only Field

Models	Box Model
--------	-----------

5.9 Box Model

Box -- *an independent (top-level) model*

A Box defines a 2-dimensional box (x, y, width, height). All coordinates are floats, which can be in any unit you want, because they're all relative to one another.

These models have a field that is a Box model :
Location.

The key field for this model is specification

specification -- *a Computed_String field of model Box*
String representation of the box formatted as: "1x2+3+4" Where '1' is the width, '2' is the height, '3' is the x coordinate, and '4' is the y.
Default: None -- this is a key field

x -- *a Number field of model Box*
Horizontal position of the upper left corner of the box.
Default: 0

y -- *a Number field of model Box*
Vertical position of the upper left corner of the box.
Default: 0

width -- *a Number field of model Box*
Width of the box
Default: 0

height -- *a Number field of model Box*
Height of the box
Default: 0

area -- *a Number field of model Box*
Area of the box
Properties: Export-Only Field

mid_x -- *a Number field of model Box*
Horizontal position of the center of the box
Default: 0

Models	Box Model
--------	-----------

mid_y -- *a Number field of model Box*
Vertical position of the center of the box
Default: 0

empty -- *a Logical field of model Box*
TRUE if the box has zero area
Properties: Export-Only Field

unspecified -- *a Logical field of model Box*
TRUE if the box is unspecified
Properties: Export-Only Field

5.10 Calendar Model

Calendar -- *on independent (top-level) model*

Calendar models daily patterns (e.g., 08:00 to 16:00 each day) that themselves may repeat in patterns related to calendar years, months, weeks, or days. For example, the 08:00 to 16:00 pattern may repeat for all weekdays (M-F), or for all third Tuesdays of each month.

Each pattern can have an associated value. For example, a Number representing the efficiency of 80% may be associated with 08:00 to 16:00 on weekdays, or a Symbol that is a setup name may be associated with the second week of each month. Each field that is of type Calendar will demand a particular 'entry_value' (the type of the value(s) in each of its 'entries'). It is an error to use a Calendar with the wrong 'entry_value'.

Each Calendar_Entry in a Calendar specifies a value applicable for a certain pattern of Dates. Those Date patterns may overlap. In that case, the Calendar_Entry with the higher 'rank' is used. That allows you to specify 08:00 to 16:00 on all weekdays (very simple to do) and then specify an overriding Calendar_Entry for Thanksgiving and the rest of your holidays. That is much easier than if you had to put the holiday exceptions into each and every Calendar_Entry.

Furthermore, one Calendar can be built up from other Calendars. The other Calendars' 'entries' are brought into the Calendar, with their 'rank's adjusted according to an Expression. In that way, several different Calendars could be defined, and each of those could use the same "holiday" Calendar to override the holidays.

There is one special uneditable Calendar named [unspecified] that has no 'entries' and no sub_calendars, nor can any entries, or sub_calendars be added to the [unspecified] calendar. The [unspecified] calendar has no values.

The model has selectors:
entry_value.

The Calendar model has these submodels:
Calendar_Entry, Sub_Calendar.

The Calendar model has fields that references these models:
Calendar_Entry, Sub_Calendar.

These models have a field that is a Calendar model:

Calendar_Entry, Sub_Calendar, Effective_Calendar_Operation, PRODUCE_YIELD_CALENDAR, PRODUCE_YIELD_RAMP_CALENDAR, DELAY_ONLY_CALENDAR, BASIC_CALENDARS, SETUP_CALENDAR, SKULL_CALENDAR, TRANSIT_CALENDAR, CALENDAR, RAMP_CALENDAR, SETUP_CALENDAR, BLOCK_CALENDAR, CALENDAR, CALENDAR_COUNT, CALENDAR_QUANTITY, CALENDAR_RATE, CALENDAR, CALENDAR, PRODUCING_FLOW_CALENDAR, PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK, SUPPLY_CALENDAR, ON_HAND_CALENDAR, ON_HAND_CALENDAR_FILTER_AND_RANK, FLOW_LIMIT_CALENDAR, FLOW_LIMIT_CALENDAR_FILTER_AND_RANK, CALENDAR, Calendar_Plan.

The key field for this model is name

This model may be extended with user-defined fields.

name -- *a Symbol field of model Calendar*

The name of this Calendar.

Default: None -- this is a key field

description -- *a String field of model Calendar*

A description of this Calendar.

Default: none

entry_value -- *an extension selector of model Calendar*

This extension selector dictates the 'value' extension of all Calendar_Entry's within this Calendar, which specifies the value(s) that are held in each entry.

For example, NUMBER has a single Number in each entry, such as might be used to represent a Calendar of Efficiencies. QUANTITY has a single Quantity in each entry, as might be used to represent replenishment supply to a Buffer.

NUMBER_QUANTITY has a Number and a Quantity in each entry. SYMBOL has a Symbol in each Entry, as might be used in a Calendar of allowed setups.

Default: UNSPECIFIED

Extensions:

UNSPECIFIED, NUMBER, QUANTITY, NUMBER_QUANTITY, SYMBOL, TIME.

entries -- *a list of Calendar_Entry submodels of model Calendar*

The individual entries that make up this Calendar.

entry (Date) -- *a Calendar_Entry field of model Calendar*
Returns the Calendar_Entry (from 'entries' or 'sub_calendars') that has the highest rank at the given Date.
Properties: Export-Only Field

dates (Date_Range) -- *a List(Date) field of model Calendar*

This returns a List(Date) at which the 'entry' of the Calendar changes during the specified Date_Range. This can be used in conjunction with 'entry' to compute the list of Calendar_Entry's that are in effect during that Date_Range.
Properties: Export-Only Field

sub_calendars -- *a list of Sub_Calendar submodels of model Calendar*
The Calendars upon which this one is built. The 'entries' in each of the 'sub_calendars' are incorporated into this Calendar, with 'rank' modified according to 'rank_expression'.

As an example of the usage, consider various Calendars created for various weekday shift patterns. Each of those are most easily specified by giving the shift hours and applying that to every weekday (Monday through Friday). However, all of those Calendars need to account for holidays. Although it is a Thursday, most of those shifts will probably not run on Thanksgiving Day. Rather than duplicate the holiday entries in each of those shift Calendars, one Calendar can be created with the holidays, and all the other Calendars can simply specify the holiday Calendar as a sub_calendar. By making the holiday Calendar_Entry's 'rank' higher than the shift Calendar_Entry's, the holiday entries will be used instead on the days they apply.

ALL CALENDARS IN A CALENDAR/SUB_CALENDAR HIERARCHY MUST HAVE THE SAME 'entry_value' extension.

5.10.1 Calendar_Entry Model

Calendar_Entry -- *a submodel of model Calendar*

Each Calendar_Entry represents a particular value that is valid during a time range each day that it applies, and the 'day_pattern' of days that it does apply during the Date_Range in which it is effective. For example, a Calendar_Entry could be effective during 1995 that specifies 80% from 8:00 to 16:00 on every weekday.

Further, each Calendar_Entry can specify a 'rank' which is used to resolve overlapping Calendar_Entry's. For example, another Calendar_Entry could specify 0% from 0:00 to 24:00 on December 25. If December 25 is a weekday in 1995, then it and the 80% entry above would both apply. In that case, the entry with the higher 'rank' would override.

Finally, each Calendar_Entry can specify a 'charge'. Such an entry is not used by default, but is instead an alternative. Using that alternative has an associated cost, as modeled by the 'charge'.

The model has selectors:
value, day_pattern.

The Calendar_Entry model has fields that references these models :
Calendar.

These models have a field that is a Calendar_Entry model :
Calendar, Calendar_Plan.

The key field for this model is name
This model may be extended with user-defined fields.

name -- *a Symbol field of model Calendar_Entry*
The name of this Calendar_Entry. Calendar_Entry names must be unique within the scope the owning calendar.
Default: None -- this is a key field

value -- *an extension selector of model Calendar_Entry*
This specifies the type of value(s) held by this Calendar_Entry. This is the same as the 'owner' Calendar's 'entry_value' extension. It is not settable here.

For example, NUMBER has a single Number in each entry, such as might be used to represent a Calendar of Efficiencies. QUANTITY has a single Quantity in each entry, as might be used to represent replenishment supply to a Buffer.

NUMBER_QUANTITY has a Number and a Quantity in each entry. SYMBOL has a Symbol in each Entry, as might be used in a Calendar of allowed setups. TIME has a Time in each Entry.

Default: NUMBER

Properties: Export-Only Field

Extensions:

NUMBER, QUANTITY, NUMBER_QUANTITY, SYMBOL, TIME.

description - - a String field of model Calendar_Entry

Comment about this particular entry (e.g., "Joe is on vacation")

Default: the empty String

effective - - a Date_Range field of model Calendar_Entry

The period during which this Calendar_Entry may be applicable. The Calendar_Entry may not be active before, nor after, the effective period. The effective period does not specify the time a Calendar_Entry will be effective, it specifies the period of time during which a Calendar_Entry may be effective. Whether, or not a Calendar_Entry is actually applicable on a given day depends on the 'day_pattern' that is used, the 'daily_start' and 'daily_end' times, and the Calendar_Entry's rank.

For example, assume a Calendar entry uses an EVERYDAY day pattern, has an effective period of 97-01-10 00:00 / 97-01-15 00:00, a 'daily_start' of 18:00, and a 'daily_end' of 26:00, or 02:00. The entry cannot start on 97-01-09 18:00 because that time is outside the effective period. The soonest it can be started is 97-01-10 00:00, which is the first second of the effective period. Similarly, The calendar entry cannot end at 97-01-16 02:00, because that time is outside the effective period. The longest it can run is 97-01-15 24:00.

The 'day_pattern', 'daily_start', and 'daily_end' define a repeating pattern. Only that part of the pattern in the effective period is visible. In essence this is an logical AND operation between the repeating pattern and the effective period.

'effective_start' specifies the earliest time a Calendar_Entry may be applicable.

'effective_end' specifies the time a Calendar_Entry can no longer be applicable. The second prior to 'effective_end' is the last second the Calendar_Entry may be applicable.

Default: forever

daily_start - - a Time field of model Calendar_Entry
For each day on which a Calendar_Entry is applicable, 'daily_start' is the time that the value specified goes into effect.

If 'daily_start' is < 0, it specifies a start time on a prior day. If 'daily_start' is >= 24:00, it specifies a start time on a subsequent day.

See documentation in 'daily_end' for more details and examples.

Default: 00:00:00

daily_end - - a Time field of model Calendar_Entry

For each day on which a Calendar_Entry is applicable, 'daily_end' is the time that the value is no longer in effect.

If 'daily_end' is < 0, it specifies an end time on a prior day. If 'daily_end' is >= 24:00, it specifies an end time on a subsequent day.

If 'daily_end' is less than 'daily_start', it specifies an end time that will occur after 'daily_start'. For example, if 'daily_start' is "18:00" and 'daily_end' is "02:00", then the actual times are "18:00" and "26:00", or from 6:00pm until 2:00am on the next day.

Both 'daily_start' and 'daily_end' are Time values. A time < "00:00" specifies a 'daily_start', or 'daily_end', on a prior day. A time >= "24:00" specifies a 'daily_start', or 'daily_end', on a subsequent day. This allows a Calendar_Entry to specify events which precede, or follow, the day on which a Calendar_Entry is applicable.

Examples:

An 8-hour work shift that begins at 10:00pm can be specified using a 'daily_start' "22:00", and a 'daily_end' of "30:00", or "06:00".

To specify a Christmas holiday from 97-12-24/97-12-25, a Calendar_Entry could be specified that is applicable on 12-25 which has a 'daily_start' of "24hr" and a 'daily_end' of "24hr". The actual start is 97-12-24 00:00 and the actual end is 97-12-25 24:00.

To specify a Thanksgiving holiday on the 4th Thursday in November, and the following day, a Calendar_Entry that is applicable on the 4th Thursday could have a 'daily_start' of "00:00" and a 'daily_end' of "48:00", which ends on Friday at "24:00".

In both the examples above, the day before Christmas and the day after Thanksgiving could have been specified using separate calendar entries. One using a DAY_OF_MONTH day pattern for 12-24, and the other using a DAY_OF_WEEK_OF_MONTH day pattern for the fourth Friday in November. However, this requires two separate calendar entries when only one is needed.

As a final example, consider the 3-day weekend of Labor Day which falls on the first Monday in September. This can be specified using the

DAY_OF_WEEK_OF_MONTH day pattern for the first Monday in September, with a 'daily_start' of "48hr" and a 'daily_end' of "24hr". This is much easier than adding two more entries for the days preceding the first Monday in September, and it even works when Labor Day falls on September 1.

Default: 24:00:00

day_pattern -- *an extension selector of model.Calendar_Entry*

This and the related fields define the days that this Calendar_Entry applies. For example, it may specify that it applies on weekdays, on M,W,F, on the third Tuesday of any month, on the fourth Thursday in November, on the 25th day of December, on the 24th of December only if it is a Thursday, or just on 91-07-23.

Default: EVERYDAY

Extensions:

EVERYDAY, EVERY_N_DAYS, WEEKDAYS, WEEKENDS, DAYS_OF_WEEK, DAY_OF_MONTH, DAY_OF_WEEK_OF_MONTH, DAY_OF_LAST_WEEK_OF_MONTH, DAY_OF_YEAR, YEARLY.

rank -- *a Number field of model.Calendar_Entry*

For days on which multiple Calendar_Entry's apply, the entry with the highest rank is used. Thus, this field allows an entry for the 25th day of December to override an entry for weekdays (normal workshift overridden by a holiday).

Default: 1

charge -- *a Quantity field of model.Calendar_Entry*

If this is non-zero, then there is a cost associated with using this Calendar_Entry. For example, running overtime will have a charge. Coupled with the rank field, you could for example set up an entry that overrides a holiday (high rank), but has a non-zero charge, and thus is only used when necessary.

Default: 0

owner -- *a Calendar field of model.Calendar_Entry*

Properties: Export-Only Field

5.10.2 Sub_Calendar Model

Sub_Calendar -- *a submodel of model.Calendar*

The Calendars upon which this one is built. The entries in each of the sub_calendars are incorporated into this Calendar, with 'rank' modified according to 'rank_expression'.

As an example of the usage, consider various Calendars created for various weekday shift patterns. Each of those are most easily specified by giving the shift hours and applying that to every weekday (Monday through Friday). However, all of those Calendars need to account for holidays. Although it is a Thursday, most of those shifts will probably not run on Thanksgiving Day. Rather than duplicate the holiday entries in each of those shift Calendars, one Calendar can be created with the holidays, and all the other Calendars can simply specify the holiday Calendar as a sub_calendar. By making the holiday Calendar_Entry's rank's higher than the shift Calendar_Entry's, the holiday entries will be used instead on the days they apply.

ALL CALENDARS IN A CALENDAR/SUB_CALENDAR HIERARCHY MUST HAVE THE SAME entry_value extension.

The Sub_Calendar model has fields that references these models:
Calendar

These models have a field that is a Sub_Calendar model:
Calendar

The key field for this model is calendar

calendar -- *a Calendar field of model.Sub_Calendar*

Each of the entries of 'calendar' and its subcalendars will be incorporated into the 'owner' Calendar. This allows, for example, a holiday Calendar to be defined once with high-ranking entries, and then reused by each of the other Calendars.

Default: None -- this is a key field

rank_expression -- *a Expression field of model.Sub_Calendar*

For each Calendar_Entry included from calendar, the rank is reset to the value returned from this Expression. The variable 'rank' is defined for this Expression to be the original 'rank' of that Calendar_Entry.

Models	Sub_Calendar Model
--------	--------------------

This allows the ranking schemes of two separately defined Calendars to be adjusted to fit together, and allows them to be incorporated into the ranking scheme of the 'owner' Calendar.

Default: rank

owner - - *a Calendar field of model Sub_Calendar*

Properties: Export-Only Field

Models	Calendar_Plan Model
--------	---------------------

5.11 Calendar_Plan Model

Calendar_Plan -- *an independent (top-level) model*

Calendar_Plan has two primary purposes:

- allow you to iterate through the Dateline (the list of resulting entries)
- allow you to select entries with cost that you want to use

The model has selectors:
entry_value.

The Calendar_Plan model has fields that references these models :
Calendar, Calendar_Entry.

The key field for this model is calendar

calendar - - *a Calendar field of model Calendar_Plan*
The Calendar that defines this Calendar_Plan.

This is not directly settable. It is generally specified by a model associated with the model that provides the Calendar_Plan field. See the documentation for the Calendar_Plan field of interest.
Default: None -- this is a key field
Properties: Export-Only Field

horizon - - *a Date_Range field of model Calendar_Plan*
The range of Dates for which the Calendar 'entries' are used to determine the values of this Calendar_Plan. Before and after this 'horizon', a single value will be used, as dictated by the model which contains the Calendar_Plan.

This is not directly settable. It is generally specified by a model associated with the model that provides the Calendar_Plan field. See the documentation for the Calendar_Plan field of interest.
Properties: Export-Only Field
entry_value - - *an extension selector of model Calendar_Plan*
This 'entry_value' extension defines the fields which specify the value to be used before and after the horizon.

Note that the 'calendar' must have the same 'entry_value' as this, or have UNSPECIFIED 'entry_value'.

This is not settable. It is dictated by the model that provides the Calendar_Plan field. See the documentation for the Calendar_Plan field of interest.

Properties: Export-Only Field

Extensions:

NUMBER, QUANTITY, NUMBER, QUANTITY, SYMBOL, TIME.

entry (Date) -- a Calendar_Entry field of model Calendar_Plan

Returns the Calendar_Entry (from 'entries' or 'sub_calendars') that has the highest rank at the given Date.

Properties: Export-Only Field

dates (Date_Range) -- a List(Date) field of model Calendar_Plan

This returns a List(Date) at which the 'entry' of the Calendar changes during the specified Date_Range. This can be used in conjunction with 'entry' to compute the list of Calendar_Entry's that are in effect during that Date_Range.

Properties: Export-Only Field

6 Control Model

Control -- an independent (top-level) model

7 Connection Model

Connection -- *an independent (top-level) model*

These models have a field that is a Connection model :
User.

7.1 User Model

User -- *an independent (top-level) model*

A User is a model of a human user of the system, or a group ("family") of users. User-specific settings are provided by this model. In particular, User security settings and User-specific directories for Reports, Layouts, Worksheets, and Styles are specified here. Also User-specific Event bindings and User-specific Command definitions are provided by this model.

The User model has these submodels :
Data_Directory, Report_Directory, Report, Layout, Worksheet, Style, Format, Domain.

The User model has fields that references these models :
User, Data_Directory, Report_Directory, Layout, Worksheet, Style, Format, Domain, Connection.

These models have a field that is a User model :
User, Data_Directory, Report_Directory, Format.

The key field for this model is name
This model may be extended with user-defined fields.

name -- *a Symbol field of model User*
The User's name or system account, by which the security is enforced.
Default: None -- this is a key field

full_name -- *a String field of model User*
The User's full name.
Default: 'name'

responsibility -- *a String field of model User*
The User's responsibility or position.
Default: none

remark -- *a String field of model User*
Often used to describe what a User is doing, where a User is, or other useful cooperative information.
Default: none

organization -- *a User field of model User*
The User organization to which this User belongs.
Default: none

members -- *a List[User] field of model User*
If this User is an organization, then this List contains each User that specifies this User as its organization.
Properties: Export-Only Field

member_of (User) -- *a Logical field of model User*
Returns "true" if this User or any of its organization's is the parameter User. More precisely, if this User is the parameter User, then it returns "true"; otherwise, if this User's organization is nonexistent, then it returns "false"; otherwise, it returns organization.member_of(parameter).
Properties: Export-Only Field

super_user -- *a Logical field of model User*
The user who starts the engine is the "super User" and has full privileges. If this is "true", then this User is the "super User".
Properties: Export-Only Field

data_directories -- *a list of Data_Directory submodels of model User*
The directories where data Worksheet definitions can be found and saved. Typically, text files to be imported or exported also reside here.

current_data_directory -- *a Pathname field of model User*
The current data worksheet directory
Default:

load_data_layouts (String) -- *a Void field of model User*
Loads the import files in the given directory. These can then be obtained from the import_files field.
Properties: command=True Export-Only Field

inc_new_data_layout (String, String) -- *a Void field of model User*
create and inc a new import file
Properties: command=True Export-Only Field

report_directories -- *a list of Report_Directory submodels of model User*
The directories where Report, Layout, Worksheet, Format and Style definitions can be found and saved.

reports -- *a list of Report submodels of model User*
The Layouts defined in this User's report_directories. Layouts define the layout of information computed in a Worksheet. The same Worksheet can be displayed in numerous different tabular forms, as a bar chart, a line chart, a Gantt chart, and so on.

The Reports defined in this User's report_directories. Reports can be used interactively (in GUI windows), printed to paper, or can generate files.

layouts -- *a list of Layout submodels of model User*
The Layouts defined in this User's report_directories. Layouts define the layout of information computed in a Worksheet. The same Worksheet can be displayed in numerous different tabular forms, as a bar chart, a line chart, a Gantt chart, and so on.
worksheets -- *a list of Worksheet submodels of model User*
The Worksheets defined in this User's report_directories. Worksheets are the computational elements of Reports.

styles -- *a list of Style submodels of model User*
The Styles defined in this User's report_directories. Styles define the "look" of Controls and Layouts. They specify colors, borders, alignment, and fonts.

formats -- *a list of Format submodels of model User*
The Formats defined in this User's report_directories. Formats define how values are converted to and from Strings.

domains -- *a list of Domain submodels of model User*
The report domains defined in this User's main report.

load_files -- *a Logical field of model User*
Loads the report files, layout files, and worksheet files owned by this user. Returns "true" if the load appears successful. If "false", the User may lack permissions or there may be some other file system problem.
Properties: command=True Export-Only Field

connections -- *a List[Connection] field of model User*
Return the list of connections used by this user
Properties: Export-Only Field

active_ui -- *a Connection field of model User*
Return the last connection created for this user

Connection Model	User Model
------------------	------------

Properties: Export-Only Field

language - - *a Symbol field of model User*
The Language preferred by this User. All text that has been translated to the specified language will be displayed translated.
Default: English

Connection Model	Report_Directory Model
------------------	------------------------

7.1.1 Report_Directory Model

Report_Directory - - *a submodel of model User*

The directories where Report, Layout, Worksheet, Format and Style definitions can be found and saved.

The Report_Directory model has fields that references these models :
User.

These models have a field that is a Report_Directory model :
User.

The key field for this model is sequence

sequence - - *a Number field of model Report_Directory*
The smaller the number, the earlier it is searched.
Default: None - - this is a key field

directory - - *a Pathname field of model Report_Directory*
The file system pathname for a directory that contains Report, Layout, Worksheet, and/or Style definitions. If this is set, then 'include' will be set to [unspecified].
Default: none

include - - *a User field of model Report_Directory*
Use the include directories of the specified user. If this is set, then 'directory' will be unspecified.
Default: [unspecified]

description - - *a String field of model Report_Directory*
A description of the kinds of Reports saved in this directory. In particular, whether or not this directory is shared with other users is important.
Default: none

editable - - *a Logical field of model Report_Directory*
If "true" then the 'owner' User has permission to edit this directory. This cannot be set "true" if the User's system account does not have permission to modify the directory. It can be set "false" either way.
Default: the native file system permissions.

owner - - *a User field of model Report_Directory*

<i>Connection Model</i>	<i>Layout Model</i>
-------------------------	---------------------

7.1.3 Layout Model

Layout -- a submodel of model User

These models have a field that is a Layout model :
User.

<i>Connection Model</i>	<i>Worksheet Model</i>
-------------------------	------------------------

7.1.4 Worksheet Model

Worksheet -- a submodel of model User

These models have a field that is a Worksheet model :
User, Data, Directory.

Connection Model	Style Model
------------------	-------------

7.1.5 Style Model

Style -- a submodel of model User

These models have a field that is a Style model :
User.

Connection Model	Format Model
------------------	--------------

7.1.6 Format Model

Format -- a submodel of model User

A Format defines how to format a value of Type 'type' into a String. It also define how to parse a String back into a value of 'Type' type. Formats are used primarily to effect the interactive display of values in Reports. For consistency, most Reports use the same named Formats; and that allows a User to override the definition of a named Format and affect the formatting consistently throughout the Reports.

The model has selectors:
spec.

The Format model has fields that references these models :
User.

These models have a field that is a Format model :
User.

The key field for this model is name_type

name_type -- a Symbol field of model Format

The name of this Format underscore the type for the format. Typically the base-name of the file system filename where this Format is saved. For example, the default date format will have a name_type of "default_date". The name field will be default; and the 'spec' field will be time;

Default: None -- this is a key field

name -- a Symbol field of model Format

The name of this format without the date-type appended to it. For example, if the Format.name_type is 'short_date_range', then format.name will be 'short', and format.spec will be 'date_range';

Default: Name from the name_type field

directory -- a Pathname field of model Format

The Pathname of the native file system directory where this Format's definition is saved.

Default: the Pathname of the directory where it was originally found

description -- a String field of model Format

A description of this Format: what it computes, how it can be used, where it is intended to be used.

Connection Model	Format Model
------------------	--------------

Default: none

editable - - *a Logical field of model* Format
If "true", then this Format can be edited.

Properties: Export-Only Field

save, save (Pathname) - - *a Logical field of model* Format

Saves this Format into 'directory'. Returns "true" if the save appears successful. If "false", the User may lack permissions or there may be some other file system problem.

Properties: Export-Only Field

handles (Type) - - *a Logical field of model* Format

Returns "true" if this Format can format and parse values of the specified Type.

Properties: Export-Only Field

spec - - *an extension selector of model* Format

Defines the type of values that can be formatted and the fields that will specify how to format those values into Strings. Those fields also affect the parsing of Strings into values.

Default: Void

Extensions:

Void, Logical, String, Symbol, Date, Date_Range, Number, Percentage, Integer, Quantity, Quantity_Range, Time, Restriction, Horizon_Date, List.

owner - - *a User field of model* Format

Properties: Export-Only Field

Connection Model	Domain Model
------------------	--------------

7.1.7 Domain Model

Domain - - *a submodel of model* User

These models have a field that is a Domain model :
User.

7.2 Model_Type Model

Model_Type -- *an independent (top-level) model*

A Model_Type models a model. Each model can be used as a value of a particular Type, a Model_Type, in Expressions. The Model_Type model describes that particular Type and the Fields that it provides.

The primary uses for the Model_Type model are to provide help documentation and to allow user-defined Fields to be added to extensible Model_Types. Note that you can also add user-defined Model_Types.

The Model_Type model has these submodels :
Field, Extension_Selector.

The Model_Type model has fields that references these models :
Model_Type, Field, Extension_Selector.

These models have a field that is a Model_Type model :
Model_Type, Field, Extension_Selector.

The key field for this model is name
This model may be extended with user-defined fields.

name -- *a Symbol field of model Model_Type*
The name of the Model_Type, which is also a Type name in the Expression language. This is only editable if user_defined is "true".
Default: None -- this is a key field

description -- *a String field of model Model_Type*
A description of the Model_Type named 'name'. This is only editable if 'user_defined' is "true".
Default: none

owner_model -- *a Model_Type field of model Model_Type*
The Model_Type of this Model_Type's 'owner' field. This Model_Type is a submodel of 'owner_model'. Top-level Model_Type will return the Void Model_Type. This is only editable if user_defined is "true".
Default: none
Properties: Export-Only Field

extensible -- *a Logical field of model Model_Type*
Returns "true" if user-defined fields may be added to 'fields' of this Model_Type. Some small, numerous models do not support the addition of user-defined fields to avoid the small overhead required (which becomes significant in large numbers). This is always "true" if user_defined is "true".
Properties: Export-Only Field

access -- *a Expression field of model Model_Type*
A logical expression that returns "true" if the current user can see the fields in this model. The 'access' Expression is passed '#' (equal to this model). The current user is available via the user' worksheet function.
Default: True

values (Typed_Value) -- *a List(Typed_Value) field of model Model_Type*
Given an instance of this model's owner, return the list of all . instances with that owner. This can be used to access the "instance" field of the model; to get all the values. Example usage:
typed_value.type.values(typed_value.owner)
Properties: Export-Only Field

fields -- *a list of Field submodels of model Model_Type*
The Fields provided by this Model_Type. Each is a function in the Expression language that can be called on a model of this Model_Type. Fields can only be added if extensible is "true".

extension_selectors -- *a list of Extension_Selector submodels of model Model_Type*
Each of these can add a different extension to a model, effectively adding some new fields to that model.

7.2.1 Field Model

Field -- *a submodel of model Model_Type*

A Field is a function that takes a model as a first argument. It returns one attribute of or the result of an analysis on that model. The Model_Type of that model determines what Fields are defined.

User-defined Fields can be added to many of the Model_Types. Only user-defined Fields may be removed. Note, however, that all data stored in that Field of a model disappears when it is removed.

The model has selectors:
field_type.

The Field model has fields that references these models :
Extension_Selector, *Model_Type*.

These models have a field that is a Field model :
Model_Type, *Field_Error*.

The key field for this model is name
This model may be extended with user-defined fields.

name -- *a Symbol field of model Field*
The name of the Field, and thus the name of the function available in Expressions.
This is only editable if user_defined is "true".
Default: None -- this is a key field

type -- *a Type field of model Field*
The Type of value returned by this Field. This is only editable if user_defined is "true".
Default: String

description -- *a String field of model Field*
A description of this field.
Default: none

default -- *a Computed_String field of model Field*
If a user defined field is not in a model, the field accessor returns NONEXISTENT.
this field gives the ability to specify some other default.

Default: NONEXISTENT
Properties: java_method=default_value

editable -- *a Logical field of model Field*
TRUE if the field can be set, FALSE if it's read-only.
Properties: Export-Only Field

field_type -- *an extension selector of model Field*
Defines the kind of field this is: SIMPLE, SELECTOR, EXTENDED, USER
Default: SIMPLE
Properties: Export-Only Field

Extensions: Export-Only Field
SIMPLE, SELECTOR, EXTENDED, USER.

access -- *a Expression field of model Field*
A logical expression that returns "true" if the current user can set this field The 'access' Expression is passed '#' (equal to this field's model). The current user is available via the user 'workspace' function.
Default: True

after_set -- *a Expression field of model Field*
An expression to execute after setting this field. The Expression in the is passed '#' (equal to this field's model). This can be useful for propagating changes in one field to other fields.
Default: none

extension_selector -- *a Extension_Selector field of model Field*
The field which can be set to one of the extensions' adding this Field to the model. If none, then this Field is always present in the model (it is not an extension Field).
Properties: Export-Only Field

extensions -- *a List(Symbol) field of model Field*
The extensions which add this Field to the model. If 'extension_selector' is set to one of the Symbols in this List, then this field is added to the model.
Properties: Export-Only Field

user_defined -- *a Logical field of model Field*
If "true", then this is a user-defined Field. Only user-defined Fields may be edited. If this returns "false", then none of the fields of this model may be edited.
Properties: Export-Only Field

value (Typed_Value) -- *a Typed_Value field of model Field*
Return the value for this field, given an instance of this field's model
Default: none
Properties: java_method=field_value

owner -- *a Model_Type field of model Field*
The Model_Type to which this Field belongs. The first argument to the Field function is a model of 'owner' Model_Type.
Properties: Export-Only Field

7.2.2 Extension_Selector Model

Extension_Selector -- *a submodel of model Model_Type*

An Extension_Selector is a special field that can be set to one of the 'extensions', which adds fields to the model.

The Extension_Selector model has fields that references these models :
Model_Type.

These models have a field that is a Extension_Selector model :
Model_Type, Field.

The key field for this model is name

name -- *a Symbol field of model Extension_Selector*
The name of the Field, and thus the name of the function available in Expressions. This is only editable if 'user_defined' is "true".
Default: None -- this is a key field
Properties: Export-Only Field

description -- *a String field of model Extension_Selector*
A description of this Field. This is only editable if 'user_defined' is "true".
Default: none

extensions -- *a List(Symbol) field of model Extension_Selector*
The extensions which can be selected.
Properties: Export-Only Field

selected_fields (Symbol) -- *a List(Field) field of model Extension_Selector*
All extension Fields added by a given extension value.
Properties: Export-Only Field

owner -- *a Model_Type field of model Extension_Selector*
The Model_Type to which this Field belongs. The first argument to the Field function is a model of 'owner' Model_Type.
Properties: Export-Only Field

7.3 Field_Error Model

Field_Error -- an independent (top-level) model

When an error occurs while editing fields or importing data, it is recorded in this model. When the error is fixed, this model is automatically removed. In this way, erroneous data is never put into the model, but it is also not lost or forgotten. In lieu of erroneous data, default non-erroneous data will be substituted. The Field_Error informs the user where that has been done so that the information can be fixed later.

The Field_Error model has fields that references these models :
Field.

The key field for this model is description

target_model -- a Typed_Value field of model Field_Error

The model that had the error (the key-field of the target_field's model). The Model_Type of 'model' is the same as the 'owner_model' of Field.

Properties: Export-Only Field

target_field -- a Field field of model Field_Error

The Field which had the error.

Properties: Export-Only Field

error_value -- a String field of model Field_Error

The erroneous value entered by user or read from data-file, based upon the Type of target_field. This is 'error_input' converted to target_field.type.

Properties: Export-Only Field

error_input -- a String field of model Field_Error

The erroneous input String entered by a user or read from a data-file.

Properties: Export-Only Field

description -- a String field of model Field_Error

A textual description of the error (what is wrong; what was done instead).

Default: None -- this is a key field

Properties: Export-Only Field

source -- a String field of model Field_Error

The source of the error. If the error occurred while reading a file, this includes the filename and line number. If the error occurred in an interactive Report, this includes name and cell location.

Properties: Export-Only Field

Function Model	Field_Error Model
----------------	-------------------

8 Function Model

Function -- an independent (top-level) model

Breakpoint Model	Field_Error Model
------------------	-------------------

9 Breakpoint Model

Breakpoint -- an independent (top-level) model

10 Extensions

Site
LINK
SUPPLIER
CUSTOMER
Seller
NONE
Site_Group
Product_Root
SIMPLE_FIXED_QUANTITY
SIMPLE_REQUEST
SIMPLE_FIXED_TIME
WEEKLY
DUAL_REQUEST
Product
SIMPLE_FIXED_QUANTITY
SIMPLE_REQUEST
SIMPLE_FIXED_TIME
WEEKLY
DUAL_REQUEST
AT_END
FCFS
PER_ALLOCATED
PER_COMMITTED
MEMBER_RANK
FIXED_SPLIT
SLIDING
HORIZON
NONE
FIXED
Operation
ALTERNATES_PRIMARY
ALTERNATES_PROPORTIONAL
EFFECTIVE_CALENDAR
DELAY_ONLY_FIXED

DELAY_ONLY_BASIC
BASIC_CALENDARS
FIXED_TIME
TIME_MULTIPLE
BASIC
BASIC_DELAYED
REQUEST_FIXED
REQUEST_FIXED_WITH_ANALYSIS
ROUTING
Load
RESOURCE
ONE
Load_Size
FIXED
LINEAR
Flow
CONSUME_FIXED
CONSUME_PER
PRODUCE_FIXED
PRODUCE_PER
PRODUCE_YIELD
PRODUCE_YIELD_CALENDAR
Operation_Problem_Detector
Operation_Plan
PRODUCE
CONSUME
DELIVER
RECEIVE
ALTERNATES_PRIMARY
ALTERNATES_PROPORTIONAL
EFFECTIVE_CALENDAR
DELAY_ONLY_FIXED
DELAY_ONLY_BASIC
BASIC_CALENDARS
FIXED_TIME
TIME_MULTIPLE

Extensions

Field Error Model

BASIC
BASIC_DELAYED
REQUEST_FIXED
REQUEST_FIXED_WITH_ANALYSIS
ROUTING

Resource

FIXED
CALENDAR
ZERO
FIXED
ZERO
UNLIMITED
FIXED_COUNT
FIXED_QUANTITY
CALENDAR_COUNT
MULTI_DIMENSION
SIMPLE_CONSOLIDATION
INFINITE_USE
EXCLUSIVE_USE
SHARED_USE
Resource_Skill
FIXED
CALENDAR
Resource_Plan
SIMPLE_CONSOLIDATION
INFINITE_USE
EXCLUSIVE_USE
SHARED_USE
Buffer
BUCKETED_NESTED_SORT
PRODUCING_FLOW_CALENDAR
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK
SUPPLY_CALENDAR
ON_HAND_CALENDAR
ON_HAND_CALENDAR_FILTER_AND_RANK
FLOW_LIMIT_CALENDAR

Extensions

Field Error Model

FLOW_LIMIT_CALENDAR_FILTER_AND_RANK
INFINITE
SUPPLIER
BASIC
BASIC_FILTER_AND_RANK
FIXED_QUANTITY
MULTIPLE
MULTIPLE_FILTER_AND_RANK
FIXED_QUANTITY_FENCED
FIXED_TIME
LFL_SIMPLE
LFL_BOUNDED
MLFL_BOUNDED
MANUAL
CALENDAR
Buffer_Problem_Detector
Buffer_Plan
BUCKETED_NESTED_SORT
PRODUCING_FLOW_CALENDAR
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK
ON_HAND_CALENDAR
ON_HAND_CALENDAR_FILTER_AND_RANK
FLOW_LIMIT_CALENDAR
FLOW_LIMIT_CALENDAR_FILTER_AND_RANK
BASIC
BASIC_FILTER_AND_RANK
FIXED_QUANTITY
MULTIPLE
MULTIPLE_FILTER_AND_RANK
FIXED_QUANTITY_FENCED
FIXED_TIME
Forecast
INDIVIDUAL
GROUP
Forecast_Entry
MEMBER_RANK

Site_Plan

LINK

SUPPLIER

CUSTOMER

Problem

REQUEST_NOT_PLANNED
 REQUEST_PLANNED_LATE
 REQUEST_PLANNED_EARLY
 REQUEST_PLANNED_SHORT
 REQUEST_PLANNED_EXCESS
 PROMISE_NOT_PLANNED
 PROMISE_PLANNED_LATE
 PROMISE_PLANNED_EARLY
 PROMISE_PLANNED_SHORT
 PROMISE_PLANNED_EXCESS
 ACCEPTANCE_NOT_PLANNED
 ACCEPTANCE_PLANNED_LATE
 ACCEPTANCE_PLANNED_EARLY
 ACCEPTANCE_PLANNED_SHORT
 ACCEPTANCE_PLANNED_EXCESS
 PLANNED_BEFORE_CURRENT
 UNRELEASED
 NEEDS_RELEASE
 INCONSISTENT_OPPLAN
 REQUEST_PROMISED_LATE
 REQUEST_PROMISED_EARLY
 REQUEST_PROMISED_SHORT
 REQUEST_PROMISED_EXCESS
 ITEM_PROMISE_OVERPRICED
 DELIVERY_PROMISE_OVERPRICED
 PROMISE_NOT_OFFERED
 PROMISE_NOT_ACCEPTED
 ACCEPTANCE_INCONSISTENT
 REQUEST_QUEUED
 DELIVERY_REQUEST_NOT_COORDINATED
 DELIVERY_PROMISE_NOT_COORDINATED

DELIVERY_ACCEPTANCE_NOT_COORDINATED
 NEGATIVE_ATP
 NEGATIVE_PLANNED_ATP
 OVER_COMMITTED
 OVER_CONSUMED
 UNALLOCATED_FORECAST
 SUPPLY_PLANNED_LATE
 SUPPLY_PLANNED_EARLY
 SUPPLY_PLANNED_SHORT
 SUPPLY_PLANNED_EXCESS
 SUPPLY_PROMISED_LATE
 SUPPLY_PROMISED_EARLY
 SUPPLY_PROMISED_SHORT
 SUPPLY_PROMISED_EXCESS
 UNIDENTIFIED_OP_STATE
 UNCONSOLIDATED
 UNCOORDINATED
 CONSOLIDATION_OVERSIZE
 CONSOLIDATION_UNDERSIZE
 OVERLOAD
 OVERSIZE
 BUCKET_OVERSIZE
 UNDERLOAD
 NEGATIVE_ON_HAND
 OVER_FLOW_LIMIT
 NEGATIVE_ON_HAND_AT_END
 LOT_OVER_CONSUMED
 LOT_NOT_CONSUMED
 LOT_NOT_PRODUCED
 LOT_OVER_PRODUCED
 LOW_ON_HAND
 EXCESS_ON_HAND
 EXCESS_ON_HAND_AT_END
 Active Strategy
 NO_PROBLEMS
 TARGET_ACHIEVED

RESOLVE_COUNT_EXCEEDED
MANUAL
SEQUENCE_RUN_ONCE
SEQUENCE_RUN_MULTIPLE
BEFORE_AND_AFTER
SEQUENTIAL_ALTERNATES
SEQUENTIAL_ALTERNATES_KEEP_BEST
PROPORTIONAL_RESOLVES
ORDERED_RESOLVES
PROPORTIONAL_INTERACTION
EARLIEST_PROBLEM_START
SORT_BY_EXPRESSION

Active_Goal

FEASIBILITY
MINIMIZE_PROBLEM_COUNT
MINIMIZE_PROBLEMS
MINIMIZE_LATENESS
WEIGHTED_LATENESS
WEIGHTED_SHORTNESS
MINIMIZE_SHORTNESS
MINIMIZE_COST
MAXIMIZE_PROFIT
MAXIMIZE_REVENUE

Item

STANDARD
CUSTOM
Skill
PRIMARY
PREFER_PRIMARY
EVEN
MAX_EFFICIENCY
Skill_Resource
FIXED
CALENDAR
Configuration
STANDARD

CUSTOM
Operation_State
EARLIEST
STARTED
COMPLETED
IN_FRONT
Request

FIXED

NUMBERED

Delivery_Request
BUCKETED_ALL
BUCKETED_ASAP
ON_TIME

ALL

ALL_ON_TIME

ASAP

ASAP_MONTHLY

BUCKETED_ALLOCATION

BUCKETED_ALL_MIN_PRICE

BUCKETED_MIN_PRICE_ASAP

SHIP_IN_RATIO

ON_TIME

FULL_QUANTITIES_OF_ALL_ITEMS

UNRESTRICTED

Promise

FIXED

Delivery_Promise

ON_TIME

FULL_QUANTITIES_OF_ALL_ITEMS

UNRESTRICTED

Horizon

ONE

MONTHS

WEEKS

DAYS

DATES

Delivery_Acceptance
ON_TIME
FULL_QUANTITIES_OF_ALL_ITEMS
UNRESTRICTED

Strategy
NO_PROBLEMS
TARGET_ACHIEVED
RESOLVE_COUNT_EXCEEDED
MANUAL
SEQUENCE_RUN_ONCE
SEQUENCE_RUN_MULTIPLE
BEFORE_AND_AFTER
SEQUENTIAL_ALTERNATES
SEQUENTIAL_ALTERNATES_KEEP_BEST
PROPORTIONAL_RESOLVES
ORDERED_RESOLVES
PROPORTIONAL_INTERACTION
EARLIEST_PROBLEM_START
SORT_BY_EXPRESSION
Problem_Set
REQUEST_NOT_PLANNED
REQUEST_PLANNED_LATE
REQUEST_PLANNED_EARLY
REQUEST_PLANNED_SHORT
REQUEST_PLANNED_EXCESS
PROMISE_NOT_PLANNED
PROMISE_PLANNED_LATE
PROMISE_PLANNED_EARLY
PROMISE_PLANNED_SHORT
PROMISE_PLANNED_EXCESS
ACCEPTANCE_NOT_PLANNED
ACCEPTANCE_PLANNED_LATE
ACCEPTANCE_PLANNED_EARLY
ACCEPTANCE_PLANNED_SHORT
ACCEPTANCE_PLANNED_EXCESS
PLANNED_BEFORE_CURRENT

UNRELEASED
NEEDS_RELEASE
INCONSISTENT_OPLAN
OPERATION
REQUEST_PROMISED_LATE
REQUEST_PROMISED_EARLY
REQUEST_PROMISED_SHORT
REQUEST_PROMISED_EXCESS
PROMISE_NOT_OFFERED
PROMISE_NOT_ACCEPTED
ACCEPTANCE_INCONSISTENT
REQUEST_QUEUED
REQUEST_PLAN
REQUEST_PLAN
PROMISE
PROMISE_PLAN
REQUEST_PROMISE
DELIVERY_REQUEST_NOT_COORDINATED
DELIVERY_PROMISE_NOT_COORDINATED
DELIVERY_ACCEPTANCE_NOT_COORDINATED
SUPPLY_PLANNED_LATE
SUPPLY_PLANNED_EARLY
SUPPLY_PLANNED_SHORT
SUPPLY_PLANNED_EXCESS
SUPPLY_PROMISED_LATE
SUPPLY_PROMISED_EARLY
SUPPLY_PROMISED_SHORT
SUPPLY_PROMISED_EXCESS
SUPPLY
SUPPLY_PLAN
SUPPLY_PROMISE
UNIDENTIFIED_OP_STATE
UNCONSOLIDATED
UNCOORDINATED
CONSOLIDATION_OVERSIZE
CONSOLIDATION_UNDERSIZE

OVERLOAD
OVERSIZE
BUCKET_OVERSIZE
UNDERLOAD
RESOURCE
NEGATIVE_ON_HAND
OVER_FLOW_LIMIT
NEGATIVE_ON_HAND_AT_END
LOT_OVER_CONSUMED
LOT_NOT_CONSUMED
LOT_NOT_PRODUCED
LOT_OVER_PRODUCED
LOW_ON_HAND
EXCESS_ON_HAND
EXCESS_ON_HAND_AT_END
BUFFER
Strategy_Change
MOVE_IN
MOVE_OUT
SPLIT
USE_MORE
USE_LESS
USE_ALTERNATE_OPERATION
USE_ALTERNATE_RESOURCE
USE_EFFECTIVE_ALTERNATE
INCREASE_CAPACITY
DECREASE_CAPACITY
RESIZE_MORE
RESIZE_LESS
UPSTREAM
DOWNSTREAM
Strategy_Lock
OPERATION_PLAN_RANK_RANGE
OPERATION_PLAN_RANK_EXPRESSION
Strategy_Goal
FEASIBILITY

MINIMIZE_PROBLEM_COUNT
MINIMIZE_PROBLEMS
MINIMIZE_LATENESS
WEIGHTED_LATENESS
MINIMIZE_SHORTNESS
WEIGHTED_SHORTNESS
MINIMIZE_COST
MAXIMIZE_PROFIT
MAXIMIZE_REVENUE
Calendar_Entry
NUMBER
QUANTITY
NUMBER_QUANTITY
SYMBOL
TIME
EVERYDAY
EVERY_N_DAYS
WEEKDAYS
WEEKENDS
DAYS_OF_WEEK
DAY_OF_MONTH
DAY_OF_WEEK_OF_MONTH
DAY_OF_LAST_WEEK_OF_MONTH
DAY_OF_YEAR
YEARLY
Calendar
UNSPECIFIED
NUMBER
QUANTITY
NUMBER_QUANTITY
SYMBOL
TIME
Flow_Criterion
CUSTOMER_RANK
SELLER_RANK
REQUEST_RANK

ACTUAL_OR_FORECAST
REQUEST_ISSUED
PROMISE_DUE
REQUEST_DUE
DUE_DATE
PROMISED
ENTRY_DATE
Calendar_Plan
NUMBER
QUANTITY
NUMBER_QUANTITY
SYMBOL
TIME
Format
Void
Logical
String
Symbol
Date
Date_Range
Number
Percentage
Integer
Quantity
Quantity_Range
Time
Restriction
Horizon_Date
List
Field
SIMPLE
SELECTOR
EXTENDED
USER

10.1 Site Extensions

10.1.1 role extensions of model Site

LINK -- a role extension of model Site

A LINK Site is a "link" in this supply "chain". It purchases Items from zero or more other Sites, performs Operations on Items, moving or transforming them, and supplies Items to zero or more other Sites.

LINK Sites can be modeled in detail. They consist of Operations which use Resources to transform or transfer Items between Buffers (the FLO network). They place Requests on other LINK and SUPPLIER Sites, and provide Promises to other LINK and CUSTOMER Sites.

The LINK model has these submodels:
Location, Item, Buffer, Resource, Skill, Operation, Configuration, Item_Group.

The LINK model has fields that references these models:

Location, Item, Buffer, Resource, Skill, Operation, Configuration, Item_Group.

managed -- a Logical field of model LINK

If "true", then this LINK Site is planned and managed by this model. The plans created for such a Site are assumed to be the plans it will follow.

If "false", then this LINK Site is managed elsewhere and this model is simply modeling its behavior. Such modeling is purely informational -- the plans created for such a Site cannot be assumed to be reliable. Only the Promises provided from the managers of that Site can be relied upon (hopefully, anyway).

Requests and Promises will be created between all unmanaged Sites and the managed Sites, just as would be created with SUPPLIER Sites and CUSTOMER Sites. In contrast, managed Sites will interface directly to each other. There is no sense in a planner making Requests to himself or responding with formal Promises to himself. Requests and Promises will be created for informational and reporting purposes, but they will be maintained automatically -- as if there was no Site boundary at all.

locations -- a list of Location submodels of model LINK
The Locations within this Site.

Extensions	LINK Extension
------------	----------------

top_locations -- *a List(Location) field of model LINK*

The Locations within this Site whose super_location is [unspecified]. These are the topmost Locations at this Site. This is a useful starting point for generating a Location hierarchy.

Properties: Export-Only Field

items -- *a list of Item submodels of model LINK*

The Items consumed, used, and/or produced at this Site. Requests placed on this Site must be for Items in this list that are sellable.

top_items -- *a List(Item) field of model LINK*

The Items within this Site whose family is [unspecified]. These are the topmost Locations at this Site. This is a useful starting point for generating an Item hierarchy.

Properties: Export-Only Field

buffers -- *a list of Buffer submodels of model LINK*

The Buffers to be used for managing the flow of Items at this Site.

resources -- *a list of Resource submodels of model LINK*

The Resources which model the capacity to perform Operations at this Site.

skills -- *a list of Skill submodels of model LINK*

The Skills needed to perform Operations. Resources have these Skills.

operations -- *a list of Operation submodels of model LINK*

The Operations that transform Items into other Items and/or move Items between Buffers.

configurations -- *a list of Configuration submodels of model LINK*

Should be hidden. The list of these is not relevant.

item_groups -- *a list of Item_Group submodels of model LINK*

Item_Groups group Items into hierarchies. Each Item can appear in at most one Item_Group in a hierarchy of Item_Groups. But each Item can appear in any number of independent Item_Group hierarchies.

top_item_groups -- *a List(Item_Group) field of model LINK*

This List is a subset of item_groups consisting only of Item_Groups that are the top of a Item_Group hierarchy, the Item_Groups that do not appear within any other Item_Group.

Properties: Export-Only Field

Extensions	SUPPLIER Extension
------------	--------------------

buffers_at_level(Integer) -- *a List(Buffer) field of model LINK*

A List of the Buffers at a particular Integer level. Note, site.buffers_at_level(3) is equivalent to site.buffers.filter(#level == 3), but is likely less expensive to compute.

Properties: Export-Only Field

operations_at_level(Integer) -- *a List(Operation) field of model LINK*

A List of the Operations at a particular Integer level. An operation level is defined as the min buffer level of all its consuming flow buffers.

Properties: Export-Only Field

buffer_levels -- *a List(Integer) field of model LINK*

A List of all the level Integer values used by any of buffers. It is a List containing 0, and the each Integer in order up to the largest level code of any of buffers. This List is generally used to feed the buffers_at_level function in order to generate a table of the Buffers by-level.

Properties: Export-Only Field

operation_levels -- *a List(Integer) field of model LINK*

A List of all the level Integer values used by any of operations. This List is generally used to feed the operations_at_level function in order to generate a table of the Operations by-level.

Properties: Export-Only Field

SUPPLIER -- *a role extension of model Site*

A SUPPLIER Site is not planned or modeled in detail; rather it represents the Items that can be procured from a supplier by LINK Sites, the Requests for those Items, and the Promises received from the supplier.

The SUPPLIER model has these submodels :

Item.

The SUPPLIER model has fields that references these models :

Item.

items -- *a list of Item submodels of model SUPPLIER*

The Items produced at this Site. Requests placed on this Site must be for one of these Items.

CUSTOMER -- *a role extension of model Site*

<i>Extensions</i>	<i>CUSTOMER Extension</i>
-------------------	---------------------------

A CUSTOMER Site is not planned or modeled in detail; rather, it is simply a destination for deliveries and source of Requests.

<i>Extensions</i>	<i>Seller Extensions</i>
-------------------	--------------------------

10.2 Seller Extensions

10.2.1 request_naming extensions of model Seller

NONE -- a request_naming extension of model Seller

Seller's Requests are not named.

10.3 Site_Group Extensions

10.3.1 spec extensions of model Site_Group

10.4 Product_Root Extensions

10.4.1 forecast_policy extensions of model Product_Root

SIMPLE_FIXED_QUANTITY -- a forecast_policy extension of model Product_Root

The Product is forecasted by a single quantity, the total delivered during the forecast Date_Range. The order_quantity defines the expected order size. All generated forecast Requests, except possibly the final one, will have the same Quantity which will be spread evenly through the dates covered by that Forecast_Entry. The final request may be less than order_quantity as the total number requested will be equal to the committed forecast.

No compensation is made for forecast error, nor for variance in timing. As current time advances and passes the generated forecast Requests, those forecasts are expired and eliminated.

The SIMPLE_FIXED_QUANTITY model has fields that references these models : Configuration.

order_quantity -- a Quantity field of model SIMPLE_FIXED_QUANTITY
The high end of the common requested Quantity's of this Product.
Default: oo

representative_configuration -- a Configuration field of model SIMPLE_FIXED_QUANTITY

The Configuration used to represent the production of this Product during master planning, when planning to forecast prior to knowing exactly what configurations will be requested.

Note that this forecast_policy creates forecast_requests based on a representative item, which must be specified in the Representative_configuration in order for forecast_requests to be generated properly. Failure to specify the representative_configuration's item will result in forecast_requests for the [unspecified] item of the [unspecified] site. (The only exception to this rule is the case in which the Product_Root has only one Product_Item. In this case, it is assumed that the single Product_Item is intended to be the 'representative_configuration.item' too.)
Properties: Export-Only Field

representative_quantity_per -- a Quantity field of model SIMPLE_FIXED_QUANTITY
The Quantity of 'representative_configuration.planned_per' unit of this Product.

Default: 1

rough_fence - - *a Horizon_Date field of model SIMPLE_FIXED_QUANTITY*
At and after the 'rough_fence', this forecast policy behaves as a SINGLE_REQUEST forecast policy.

Default: infinite future

SINGLE_REQUEST - - *a forecast_policy extension of model Product_Root*

The Product is forecasted by a single quantity, the total delivered during the forecast Date_Range. One forecast Request will be generated at the beginning of each forecast_period, having the same Quantity as 'committed' for that Forecast_Entry.

No compensation is made for forecast error, nor for variance in timing. As current time advances and passes the generated forecast Requests, those forecasts are expired and eliminated.

The SINGLE_REQUEST model has fields that references these models :
Configuration.

representative_configuration - - *a Configuration field of model SINGLE_REQUEST*

The Configuration used to represent the production of this Product during master planning, when planning to forecast prior to knowing exactly what configurations will be requested.

Note that this forecast_policy creates forecast_requests based on a representative item, which must be specified in the 'representative_configuration' in order for forecast_requests to be generated properly. Failure to specify the 'representative_configuration's item will result in forecast_requests for the (unspecified) item of the (unspecified) site. (The only exception to this rule is the case in which the Product_Root has only one Product_Item. In this case, it is assumed that the single Product_Item is intended to be the 'representative_configuration.item' too.)

Properties: Export-Only Field
representative_quantity_per - - *a Quantity field of model SINGLE_REQUEST*
The Quantity of 'representative_configuration' planned per unit of this Product.
Default: 1

SIMPLE_FIXED_TIME - - *a forecast_policy extension of model Product_Root*

The Product is forecasted by a single quantity, the total delivered during the forecast Date_Range. The generated forecast requests will be placed at the beginning of the bucket and then at each 'interval' of time through the rest of the bucket. The requests will be for approximately the same quantity, though may be rounded up or down to the item's 'unit'. The sum of the requests will be the forecasted quantity rounded up to the item's 'unit'.

No compensation is made for forecast error, nor for variance in timing. As current time advances and passes the generated forecast Requests, those forecasts are expired and eliminated.

The SIMPLE_FIXED_TIME model has fields that references these models :
Configuration.

interval - - *a Time field of model SIMPLE_FIXED_TIME*
The Time between the generated forecast Delivery_Requests.
Default: 1 week

representative_configuration - - *a Configuration field of model SIMPLE_FIXED_TIME*

The Configuration used to represent the production of this Product during master planning, when planning to forecast prior to knowing exactly what configurations will be requested.

Note that this forecast_policy creates forecast_requests based on a representative item, which must be specified in the 'representative_configuration' in order for forecast_requests to be generated properly. Failure to specify the 'representative_configuration's item will result in forecast_requests for the (unspecified) item of the (unspecified) site. (The only exception to this rule is the case in which the Product_Root has only one Product_Item. In this case, it is assumed that the single Product_Item is intended to be the 'representative_configuration.item' too.)

Properties: Export-Only Field
representative_quantity_per - - *a Quantity field of model SIMPLE_FIXED_TIME*
The Quantity of 'representative_configuration' planned per unit of this Product.
Default: 1

rough_fence - - *a Horizon_Date field of model SIMPLE_FIXED_TIME*
At and after the 'rough_fence', this forecast policy behaves as a SINGLE_REQUEST forecast policy.
Default: infinite future

WEEKLY -- *a forecast_policy extension of model Product_Root*

The Product is forecasted by a single quantity, the total delivered during the forecast Date_Range. The *day_of_week* defines the expected delivery day. The generated forecast requests will be placed on *day_of_week* at one week intervals of time throughout the bucket. The requests will be for approximately the same quantity, though they may be rounded up or down to the Item's unit. The sum of the requests will be the forecasted quantity rounded up to the Item's unit.

No compensation is made for forecast error, nor for variance in timing. As current time advances and passes the generated forecast Requests, those forecasts are expired and eliminated.

The WEEKLY model has fields that references these models :

Configuration.

day_of_week -- *a Integer field of model WEEKLY*

The day of each week that the Delivery_Requests should be due. Monday is day 1; Sunday is day 7.

Default: Monday

representative_configuration -- *a Configuration field of model WEEKLY*

The Configuration used to represent the production of this Product during master planning, when planning to forecast prior to knowing exactly what configurations will be requested.

Note that this forecast_policy creates forecast_requests based on a representative item, which must be specified in the 'representative_configuration' in order for forecast_requests to be generated properly. Failure to specify the 'representative_configuration's item will result in forecast_requests for the (unspecified) item of the (unspecified) site. (The only exception to this rule is the case in which the Product_Root has only one Product_Item. In this case, it is assumed that the single Product_Item is intended to be the 'representative_configuration.item' too.)

Properties: Export-Only Field

representative_quantity_per -- *a Quantity field of model WEEKLY*

The Quantity of 'representative_configuration' planned per unit of this Product.

Default: 1

rough_fence -- *a Horizon_Date field of model WEEKLY*
At and after the 'rough_fence', this forecast policy behaves as a SINGLE REQUEST forecast policy.

Default: infinite future

DUAL_REQUEST -- *a forecast_policy extension of model Product_Root*

The Product forecast demand is placed on the representative item's site via dual forecast Requests.

One forecast Request represents the amount of the Forecast_Entry's committed quantity which has not been covered by any allocation (i.e. forecast_entry.committed - forecast_entry.allocated). This forecast Request represents the amount "still needed" to satisfy the forecast.

The other forecast Request represents the amount of the Forecast_Entry's committed quantity which has been covered by an allocation but has not yet been consumed (i.e. forecast_entry.allocated - forecast_entry.consumed). This forecast Request represents the "unconsumed allocation" of the forecast_entry.

No compensation is made for forecast error, nor for variance in timing. As current time advances and passes the generated forecast Requests, those forecasts are expired and eliminated.

The DUAL_REQUEST model has fields that references these models :

Configuration.

representative_configuration -- *a Configuration field of model DUAL_REQUEST*
The Configuration used to represent the production of this Product during master planning, when planning to forecast prior to knowing exactly what configurations will be requested.

Note that this forecast_policy creates forecast_requests based on a representative item, which must be specified in the 'representative_configuration' in order for forecast_requests to be generated properly. Failure to specify the 'representative_configuration's item will result in forecast_requests for the (unspecified) item of the (unspecified) site. (The only exception to this rule is the case in which the Product_Root has only one Product_Item. In this case, it is assumed that the dual Product_Item is intended to be the 'representative_configuration.item' too.)

Properties: Export-Only Field

<i>Extensions</i>	<i>DUAL_REQUEST Extension</i>
-------------------	-------------------------------

representative_quantity_per -- *a Quantity field of model DUAL_REQUEST*
The Quantity of *representative_configuration* planned per unit of this Product.
Default: 1

rough_fence -- *a Horizon Date field of model DUAL_REQUEST*
At and after the *rough_fence*, this forecast policy behaves as a SINGLE_REQUEST
forecast policy.
Default: infinite future

<i>Extensions</i>	<i>Product Extensions</i>
-------------------	---------------------------

10.5 Product Extensions

10.5.1 forecast_policy extensions of model Product

SIMPLE_FIXED_QUANTITY -- *a forecast_policy extension of model Product*

A **SIMPLE_FIXED_QUANTITY** Product is defined by a
SIMPLE_FIXED_QUANTITY Product_Root. See the documentation there.

SINGLE_REQUEST -- *a forecast_policy extension of model Product*

A **SINGLE_REQUEST** Product is defined by a **SINGLE_REQUEST** Product_Root.
See the documentation there.

SIMPLE_FIXED_TIME -- *a forecast_policy extension of model Product*

A **SIMPLE_FIXED_TIME** Product is defined by a **SIMPLE_FIXED_TIME**
Product_Root. See the documentation there.

WEEKLY -- *a forecast_policy extension of model Product*

A **WEEKLY** Product is defined by a **WEEKLY** Product_Root. See the documentation
there.

DUAL_REQUEST -- *a forecast_policy extension of model Product*

A **DUAL_REQUEST** Product is defined by a **DUAL_REQUEST** Product_Root. See
the documentation there.

10.5.2 expiration_policy extensions of model Product

AT_END -- *a expiration_policy extension of model Product*

The forecast does not expire until the end of the forecast period. So, if 1000 units are
forecasted, but only 10 units have actually been requested through the 26th day of the
month period, the other 990 units will continue to be forecasted to be sold in the last
few days.

10.5.3 allocation_policy extensions of model Product

FCFS -- *a allocation_policy extension of model Product*

The **FCFS** allocation_policy does no allocation. Rather it makes the whole allocation
available at this Seller for the Members' to consume first-come first-served (FCFS).

PER_ALLOCATED -- a allocation_policy extension of model Product

The `PER_ALLOCATED` allocation_policy distributes initial allocations proportionally to the quantity committed 'to by that Seller. Thereafter, allocations are adjusted proportionally to the quantity 'allocated' to that Seller. A portion of the allocation can be retained for use at the discretion of this Seller. Any leftovers due to lot sizing or explicit adjustments will be available on a first-come-first-served (FCFS) basis.

retain - - a Percentage field of model PER_ALLLOCATED

The Percentage of the allocated Quantity that should be retained for use at the discretion of this Seller. No 'member' will get access to it unless the Seller explicitly reduces the 'retained' Quantity in a bucket.

Default: 0%

PER_COMMITTED -- a allocation_policy extension of model Product

The `PEE_COMMITTED` allocation policy distributes the allocations proportional to the quantity committed to by that Seller. A portion of the allocation can be "trained" for use at the discretion of this Seller. Any leftovers due to lot sizing or explicit adjustments will be available on a first-come-first-served (FCFS) basis.

retain - a Percentage field of model PER_COMMITTED

The Percentage of the allocated Quantity that should be retained for use at the discretion of this Seller. No 'member' will get access to it unless the Seller explicitly reduces the 'retained' Quantity in a bucket.

Default: 0%

MEMBER_RANK -- a allocation_policy extension of model Product

The MEMBERS_RANK allocation policy distributes the allocations to the 'members' in the order of their 'rank'. For Sellers with equal 'rank', it distributes to each proportionally to the quantity committed to by that Seller. A portion of the allocation can be 'reined' for use at the discretion of this Seller. Any leftovers due to not sizing or explicit adjustments will be available on a first-come-first-served (FCFS) basis. For example: Let us take a hypothetical situation, in which we have the following Seller hierarchy with member ranks shown in parentheses. Their forecasts are shown next to their ranks. Initial allocations for all sellers is zero. After planning, the promise is, say 2000. Now the Top Seller needs to distribute the change in the allocation (2000-0=2000) to it's members. When there is an increase in allocation at a seller, we take the member with the highest rank. In this case we have two members with equal ranks. Now distribute 2000 between Northern Sales and southern Sales proportional to their committed (1500:1000=5:2). So the Northern Sales gets 1200 and the Southern Sales

gets 800. Eastern Sales does not get anything. Now we need to distribute 1200 (1200-0) between N1 Sales and N2 Sales. When there is an increase in allocation to the Northern Sales, we take the highest ranked member, i.e. N2 Sales, and allocate 250 to him. Then we assign 250 for N1 Sales. Northern Sales keeps the rest, i.e., 700 (1200 - 250 - 250).

Now the promise was changed to, say 800. Now we need to allocate -1200 (800-2000). To the members of the Top Seller: When there is a reduction in allocation, we take the lowest ranked member and try to make his net allocation zero. So we take Eastern Sales. His allocation is already zero. So we can't give (take from) him any more. Now take the next lowest ranked seller, Northern Sales and Southern Sales. Now try to allocate -1200 to these sellers proportional to their committed. Northern Sales gets -720 and Southern Sales gets -480. So their net allocations becomes 480 (1200+/-720) and 320 (800+/-480). Now the Northern Sales need to distribute -720 between him self and his sub sellers. We try to make the allocation to the Northern Sales' forecast (1500 - (250+250) = 1000) to zero. So we assign -700 to him. So we have -20 to distribute between N1 and N2. We take the lowest ranked member and try to assign -20. So we assign -20 to N1 Sales. His net allocation becomes 230. Allocations for N2 Sales does not change.

	Top Seller
(10) 3000	
500 Southern Sales (5)	Northern Sales (5) 1500 Eastern Sales (1)

(2) 250 N2 Sales (3) 250	N1 Sales

retain - a Percentage field of model MEMBER_RANK

The Percentage of the allocated Quantity that should be retained for use at the discretion of this Seller. No Member will get access to it unless the Seller explicitly reduces the 'retained' Quantity in a bucket.

Default: 0%

pool_allocation - - a Logical field of model MEMBER_RANK

If this flag is true then allocation for the member setters whose `lock_allocated` flag is true will be pooled at the organization level. This quantity will not be available to the organization or other members of the organization. It will be reserved for the members whose `lock_allocated` is true.

Default: FALSE

reallocate (Plan) - - a *Void field of model MEMBER_RANK*

Run the allocation policy on the forecast in the specified plan ignoring the previous allocations. The allocated field of member seller is populated in the following order: First try to meet the consumed quantity. Then accepted quantity and finally committed quantity

Properties: command=True Export-Only Field

FIXED_SPLIT -- a allocation_policy extension of model Product

The FIXED_SPLIT allocation_policy distributes according a fixed percentage breakdown among the members of this Product's owner. This breakdown is defined in the allocations sub-model. All members of the Product's owner which are not assigned a fixed percentage split in the allocations list are assumed to be allocated zero.

A portion of the allocation can be retained for use at the discretion of this Seller. Any leftovers due to lot sizing or explicit adjustments will be available on a first-come-first-served (FCFS) basis.

If the sum of the fixed percentage splits defined in allocations is not equal to 100%, then the splits are normalized to 100%. Then the original amount is reduced by the retain percent and the result is allocated using the normalized splits.

The FIXED_SPLIT model has these submodels:
Product_Allocation.

The FIXED_SPLIT model has fields that references these models:
Product_Allocation.

retain - - a *Percentage field of model FIXED_SPLIT*

The Percentage of the allocated Quantity that should be retained for use at the discretion of this Seller. No member will get access to it unless the Seller explicitly reduces the retained Quantity in a bucket.

Default: 0%

allocations - - a *list of Product_Allocation submodels of model FIXED_SPLIT*
The allocations or fixed percentages which are used by this product's owner when splitting its allocated amount up among its members. Any entry of allocations whose split is set to zero is automatically deleted.

10.5.4 availability_policy extensions of model Product

SLIDING -- a availability_policy extension of model Product

The SLIDING availability policy represents the default behavior of the owner Product's consume_earlier field. This behavior allows modeling of ATP which expires over time, by establishing a sliding window of allocation availability. The width of the sliding window is defined by the consume_earlier field.

Unless otherwise stated, this SLIDING behavior is baseline functionality which is included in all other availability_policies.

HORIZON -- a availability_policy extension of model Product

The HORIZON availability policy represents the addition of a series of fixed boundaries to the baseline SLIDING functionality. The fixed boundaries are specified by setting the horizon's bucket_spec to the appropriate extension. This behavior allows modeling of ATP which both expires gradually over time (the baseline functionality defined by consume_earlier) and also expires completely at a fiscal boundary (such as month-end or quarter-end).

The HORIZON model has fields that references these models:
Horizon.

availability_horizon - - a *Horizon field of model HORIZON*

Define how the Product's availability_dates are computed. This Horizon defines the availability boundaries which will limit the accumulation of ATP for this Product.

Default: [unspecified]

buckets (Date_Range) - - a *List(Date_Range) field of model HORIZON*

This will return list of buckets defined by currently selected bucket_spec extension.

If no argument is passed, this returns list of buckets within planning horizon(plan start to plan end). Start date for the first bucket will be same as the plan start and the end date for the last bucket in the list will be same as plan end. !!!(INIMPLE-MENTED)!!!!

If Date_Range argument is given, this will return list of buckets within that Date_Range. First bucket's start will be same as the start date of the Date_Range passed in as argument. Similarly, the end date of last bucket will be same as the end date of the Date_Range passed in as argument.

Properties: Export-Only Field

<i>Extensions</i>	<i>price_policy extensions of model Product</i>
-------------------	---

10.5.5 price_policy extensions of model Product

NONE -- *a price_policy extension of model Product*

The price_policy, NONE, is used when a Product's pricing is not to be considered. All ATP prices for such a Product will be 0.

FIXED -- *a price_policy extension of model Product*

The FIXED price_policy supports fixed pricing. The price for a Product may be specified in the price field. All ATP prices for that Product will then reflect the specified price.

price -- *a Money field of model FIXED*

The base price for 1 unit of this Product.

Default: 0

<i>Extensions</i>	<i>Operation Extensions</i>
-------------------	-----------------------------

10.6 Operation Extensions

10.6.1 process extensions of model Operation

ALTERNATES_PRIMARY -- *a process extension of model Operation*

An ALTERNATES_PRIMARY Operation executes one of the sub_operations. It defaults to using the primary Operation, the one with the largest 'percentage'. When it needs to offload, it selects an alternate probabilistically, selecting proportional to 'percentage'.

The difference between ALTERNATES_PRIMARY and ALTERNATES_PROPORTIONAL is that ALTERNATES_PRIMARY always selects the one with the largest 'percentage' initially, but ALTERNATES_PROPORTIONAL makes the initial selection probabilistically also.

The ALTERNATES_PRIMARY model has these submodels :
Alternate_Operation.

The ALTERNATES_PRIMARY model has fields that references these models :
Alternate_Operation.

splittable -- *a Logical field of model ALTERNATES_PRIMARY*

This field exists only for backward compatibility. It will be implemented in a future release.

Default: false

Properties: obsolete=True

alternates -- *a list of Alternate_Operation submodels of model*

ALTERNATES_PRIMARY

The alternate Operations that may be selected to perform this Operation. Each of these alternates will show up in 'sub_operations' of this Operation, but only the selected alternate will show up in 'sub_operation_plans' of an Operation_Plan of this Operation.

ALTERNATES_PROPORTIONAL -- *a process extension of model Operation*

An ALTERNATES_PROPORTIONAL Operation executes one of the sub_operations. It chooses which alternate to use probabilistically, weighted by the

Alternate_Operation's 'percentage'.

The ALTERNATES_PROPORTIONAL model has these submodels :

Extensions	EFFECTIVE_CALENDAR Extension
------------	------------------------------

Alternate_Operation.

The ALTERNATES_PROPORTIONAL model has fields that references these models
:
Alternate_Operation.

splittable - - a Logical field of model ALTERNATES_PROPORTIONAL
This field exists only for backward compatibility. It will be implemented in a future release.

Default: false
Properties: obsolete=True

alternates - - a list of Alternate_Operation submodels of model ALTERNATES_PROPORTIONAL

The alternate Operations that may be selected to perform this Operation. Each of these alternates will show up in 'sub_operations' of this Operation, but only the selected alternate will show up in 'sub_operation_plans' of an Operation_Plan of this Operation.

EFFECTIVE_CALENDAR -- a process extension of model Operation

An EFFECTIVE_CALENDAR Operation executes one of the sub_operations. Selection of the sub operation will be done based on the effectivity specified by the calendar for that sub operation. Each sub operation can specify their own effectivity calendar. If more than one sub operation is effective on a given date, probabilistic selection will be done based on the 'percentage' value.

The EFFECTIVE_CALENDAR model has these submodels :
Effective_Calendar_Operation.

The EFFECTIVE_CALENDAR model has fields that references these models :
Effective_Calendar_Operation.

splittable - - a Logical field of model EFFECTIVE_CALENDAR
This field exists only for backward compatibility. It will be implemented in a future release.
Default: false
Properties: obsolete=True

Extensions	DELAY_ONLY_FIXED Extension
------------	----------------------------

effective_alternates - - a list of Effective_Calendar_Operation submodels of model EFFECTIVE_CALENDAR

The alternate Operations that may be selected to perform this Operation. Each of these alternates will show up in 'sub_operations' of this Operation, but only the selected alternate will show up in 'sub_operation_plans' of an Operation_Plan of this Operation.

DELAY_ONLY_FIXED -- a process extension of model Operation

A DELAY_ONLY_FIXED Operation loads no Resource. It simply runs for a fixed amount of time, independent of the quantity being processed. Thus, this can be used to insert a fixed delay in the processing. It is also commonly used to model 'fixed-lead-time Operations, which are particularly common when modeling outsourcing and purchased items.

time - - a Time field of model DELAY_ONLY_FIXED
The fixed amount of Time that this Operation runs (delays).
Default: 0

DELAY_ONLY_BASIC -- a process extension of model Operation

A DELAY_ONLY_BASIC Operation loads no Resource. It simply runs for a time equal to 'fixed_time + units*time_per'. Thus, this can be used to insert a per-unit delay in the processing.

time - - a Time field of model DELAY_ONLY_BASIC
The fixed additional Time that this Operation runs (delays), beyond the computed per-unit Time.
Default: 0

time_per - - a Time field of model DELAY_ONLY_BASIC
The Time per unit of the Operation that this Operation runs (delays).
Default: 0

BASIC_CALENDARS -- a process extension of model Operation

A BASIC_CALENDARS Operation loads the Resources for a time equal to 'time + units*time_per/resource_efficiency', where both the time per unit of the operation ('time_per') and the additional operation time ('time') are specified using calendars. In calculating the overall operation time, the amount of 'fixed time' (using 'time_calendar') is scheduled before the 'per-unit time' chronologically in the operation plan. The algorithm used will produce the same results whether the operation is planned end-backward or start-forward. Both the 'fixed time' component and the 'per-

unit time' component are calculated using a weighted average of the calendar entries specified for those times, over the required time ranges. For example, assume an operation plan's fixed time segment overlaps two calendar entries where the latter entry specifies 5 days of fixed time, and the former entry specifies 10 days of fixed time. If the operation plan overlaps the 5-day segment by three days, this comprises 60 percent of the required fixed time (3 days where 5 are required). This means that 40 percent of the required fixed time (of 100 percent) must come from the segment where 10 days of fixed time are required. So when 10 days are required, 40 percent of that is 4 days. Therefore, the total fixed time for this operation will be 7 days, using the weighted average of the calendar entries. The actual number of calendar entries used in computing fixed time and per-unit time may be greater than or less than two, depending on the particular characteristics of the plan being generated.

The BASIC_CALENDARS model has fields that reference these models:

Calendar.

time_calendar -- *a Calendar field of model BASIC_CALENDARS*

The additional standard (100% efficiency) Time that this Operation runs beyond the computed per-unit Time. This time will vary, depending on the specified calendar. The time used will be taken from the calendar on the day the operation is scheduled to begin. The time specified will be multiplied by the loaded Resources' efficiencies to determine actual time.

Default: [unspecified]

time_per_calendar -- *a Calendar field of model BASIC_CALENDARS*

The standard (100% efficiency) Time per unit of the Operation that this Operation runs, as specified in a calendar. The time specified will be multiplied by the loaded Resources' efficiencies to determine actual time.

Default: [unspecified]

FIXED_TIME -- a process extension of model Operation

A FIXED_TIME Operation loads a Resource for a fixed amount of Time, independent of the Quantity being processed. It is the same as BASIC with time_per equal to zero (except that this is smaller and faster).

time -- *a Time field of model FIXED_TIME*

The fixed time that this Operation runs. Note that this is standard time which will be multiplied by the loaded Resources' efficiencies to determine actual time.

Default: 0

TIME_MULTIPLE -- a process extension of model Operation

A TIME_MULTIPLE Operation loads a Resource for a Time that increases in discrete multiples of base_time; based on the number of units as a multiple of base_units. More mathematically, the time is $\text{round_up}(\text{units} / \text{base_units}) * \text{base_time}$.

base_time -- *a Time field of model TIME_MULTIPLE*

The base Time per base_unit that this Operation runs. Note that this is standard time which will be multiplied by the loaded Resources' efficiencies to determine actual time.

Default: 0

base_units -- *a Number field of model TIME_MULTIPLE*

The base number of units that can be run in base_time.

Default: 1

BASIC -- a process extension of model Operation

A BASIC Operation loads the Resources for a time equal to $\text{time} + \text{units} * \text{time_per} / \text{resource_efficiency}$.

time -- *a Time field of model BASIC*

The fixed additional Time that this Operation runs, beyond the computed time_per unit. Note that this is standard time which will be multiplied by the loaded Resources' efficiencies to determine actual time.

Default: 0

time_per -- *a Time field of model BASIC*

The standard (100% efficiency) Time per unit of this Operation that this Operation runs. Note that this is standard time which will be multiplied by the loaded Resources' efficiencies to determine actual time.

Default: 0

BASIC_DELAYED -- a process extension of model Operation

A BASIC_DELAYED Operation loads the Resources for a time equal to $\text{time} + \text{units} * \text{time_per_resource_efficiency}$. In addition, the BASIC_DELAYED Operation total time includes a fixed 'warm_up_delay' that occurs prior to loading the Resources and a fixed 'cool_down_delay' that occurs after loading the Resources.

Note that delay Times are not affected by the efficiency of the Resources.

Extensions	REQUEST_FIXED Extension
------------	-------------------------

time - - - *a Time field of model BASIC_DELAYED*

The fixed additional Time that this Operation runs, beyond the computed *time_per* unit. Note that this is standard time which will be multiplied by the loaded Resources' efficiencies to determine actual time.

Default: 0

time_per - - - *a Time field of model BASIC_DELAYED*

The standard (100% efficiency) Time per unit of this Operation that this Operation runs. Note that this is standard time which will be multiplied by the loaded Resources' efficiencies to determine actual time.

Default: 0

pre_load_delay - - - *a Time field of model BASIC_DELAYED*

The fixed delay Time from the start of the Operation to the time when it begins loading the Resources and running. This time occurs immediately before *time* and *time_per*.

Default: 0

post_load_delay - - - *a Time field of model BASIC_DELAYED*

The fixed delay Time from when the Operation has finished loading the Resources until the Operation ends. This time occurs immediately after *time* and *time_per*.

Default: 0

REQUEST_FIXED - - a process extension of model Operation

A REQUEST_FIXED Operation places a Request for an Item 'item' at another Site (supplier). The Operation requires time 'from completion of delivery of the item' by the supplier to completion of this Operation (when the output Flows will produce to those Buffers). Thus, this could model a simplistic pickup requirement (the 'delivery' is made to the supplier's dock from where you can pick it up). Or it could represent the time to perform inspection or receiving activity. Or it could be a simple pad on the due date given the supplier.

The REQUEST_FIXED model has fields that references these models:

Configuration, Item, Site, Seller

time - - - *a Time field of model REQUEST_FIXED*

The fixed Time that this Operation requires following delivery by the supplier.

Default: 0

Extensions	REQUEST_FIXED Extension
------------	-------------------------

order_lead_time - - - *a Time field of model REQUEST_FIXED*

Obsolete! The field *order_lead_time* is functionally covered by the 'release_fence'. To model a supplier lead time of 5 days, use a 5-day Horizon. Date for the release_fence. In that way, an unclassified REQUEST_FIXED Operation will not be planned inside the supplier lead time. Note that the release_fence, being a Horizon_Date, allows you to specify a varying supplier lead time, more accurately modelling the real constraint. And it does so with the same mechanism that is used for non-REQUEST_FIXED Operations.

Default: 0

Properties: obsolete=True

item - - - *a Item field of model REQUEST_FIXED*

The Item from site that is being requested (same as configuration.item).

To set this you must have an Item. If you have just the name of an Item, then you must also have the Site. With those two you can set this to *site.items.find(item_name)*. However, it is generally simpler to set 'supplier' to that site and 'item_name' to that name.

Default: [unspecified]

supplier - - - *a Site field of model REQUEST_FIXED*

The Site from which the Item is requested (same as 'item.owner').

However, it is also scitable. Setting this will cause 'configuration.item' to be set to the Item by the same name in the newly set Site. If the new Site does not have an Item with that name, then it is set to the [unspecified] Item at that Site.

Note that setting 'item' or 'configuration.item' to an Item also sets this 'supplier' to be the Site that owns that Item.

Default: configuration.item.owner

item_name - - - *a Symbol field of model REQUEST_FIXED*

This is the same as 'item.name', however it is also scitable. Setting 'item_name' indirectly sets 'configuration.item' to the Item with the specified name in the same Site (supplier). If there is no such Item, a Field_Error is reported and nothing is changed.

Default: configuration.item.name

seller - - - *a Seller field of model REQUEST_FIXED*

The Seller through which this Site purchases from 'supplier'. If this is left [unspecified], then the price will be assumed zero, and no forecasted demand will be consumed at the 'supplier'.

Default: [unspecified]

REQUEST_FIXED_WITH_ANALYSIS -- a process extension of model Operation

A **REQUEST_FIXED_WITH_ANALYSIS** Operation places a Request for an Item at another Site (supplier). The Operation requires time from completion of delivery of the item by the supplier to completion of this Operation (when the output Flows will supply to those Buffers). Thus, this could model a simplistic pickup requirement (the "delivery" is made to the supplier's dock from where you can pick it up). Or it could represent the time to perform inspection or receiving activity. Or it could be a simple pad on the due date given the supplier. Additionally, when the corresponding item request is created, *** add description ***

The **REQUEST_FIXED_WITH_ANALYSIS** model has fields that references these models:

Configuration, Item, Site, Seller.

time -- a Time field of model **REQUEST_FIXED_WITH_ANALYSIS**

The fixed Time that this Operation requires following delivery by the supplier.

Default: 0

order_lead_time -- a Time field of model

REQUEST_FIXED_WITH_ANALYSIS

The typical order lead Time that this Request requires.

Default: 0

Item -- a Item field of model **REQUEST_FIXED_WITH_ANALYSIS**

The Item from site that is being requested (same as configuration.item).

To set this you must have an Item. If you have just the name of an Item, then you must also have the Site. With those two you can set this to `site.items.find(item_name)`. However, it is generally simpler to set 'supplier' to that site and 'item_name' to that name.

Default: [unspecified]

supplier -- a Site field of model **REQUEST_FIXED_WITH_ANALYSIS**

The Site from which the Item is requested (same as item.owner).

However, it is also sciable. Setting this will cause 'configuration.item' to be set to the Item by the same name in the newly set Site. If the new Site does not have an Item with that name, then it is set to the [unspecified] Item at that Site.

Note that setting 'item' or 'configuration.item' to an Item also sets this 'supplier' to be the Site that owns that Item.

Default: configuration.item.owner

item_name -- a Symbol field of model **REQUEST_FIXED_WITH_ANALYSIS**

This is the same as 'item.name'; however it is also settable. Setting 'item_name' indirectly sets 'configuration.item' to the Item with the specified name in the same Site (supplier). If there is no such Item, a **Field_Error** is reported and nothing is changed.

Default: configuration.item.name

seller -- a Seller field of model **REQUEST_FIXED_WITH_ANALYSIS**

The Seller through which this Site purchases from 'supplier'. If this is left [unspecified], then the price will be assumed zero, and no forecasted demand will be consumed at the 'supplier'.

Default: [unspecified]

ROUTING -- a process extension of model Operation

A **ROUTING** Operation executes the 'sub_operations' in order, with no overlap. The 'sub_operations' and their ordering are defined by the 'routing_operations'.

It can add Loads which consume Resources for the duration of the 'sub_operations', or it can add Flows which consume Items at the beginning of the 'sub_operations' or produce Items at the end.

If Resources are loaded by this **ROUTING** Operation, then the efficiency of the Resource affects all 'sub_operations', but does not affect the time between 'sub_operations'. For example, if a Resource used by this **ROUTING** Operation drops to 50%, then the maximum efficiency of any sub Operation will be 50% (if it was at 100%, the load time required will double).

If a **ROUTING** Operation is not 'interruptible', then the Resources it loads will be reserved for the entire duration of the routing, whether or not any 'sub_operations' are being performed. The Resources must be available for that whole time (no 0% efficiency). This is often used for vats, molds, dies, mandrel, or other Resources that must move with the material through several Operations before being separated.

If a **ROUTING** Operation is 'interruptible', then the Resources it loads will only be loaded while 'sub_operations' are using it. The Resource can be unavailable at other times. Effectively, such Loads become additional Loads of the 'sub_operations'. This is often used to model a sequence of Operations that can choose among alternate Resources, but must all choose the same alternate.

Extensions	ROUTING Extension
------------	-------------------

Currently, 'interruptible' is not implemented, and all ROUTING operations are planned as though 'interruptible' is true.

Note that for a non-interruptible ROUTING Operation, a drop in efficiency on a Resource it loads may cause all the 'sub_operations' to increase in time, but the total time that the Resource is reserved may not change since the time between those 'sub_operations' may decrease.

The ROUTING model has these submodels :
ROUTING_Operation.

The ROUTING model has fields that references these models :
ROUTING_Operation.

routing_operations -- *a list of Routing_Operation submodels of model ROUTING*
This defines the 'sub_operations' of this Operation and their numbered sequence.

Extensions	Load Extensions
------------	-----------------

10.7 Load Extensions

10.7.1 usage_policy extensions of model Load

RESOURCE -- *a usage_policy extension of model Load*

A RESOURCE usage_policy Load utilizes the specified 'resource';

ONE -- *a usage_policy extension of model Load*

A ONE usage_policy Load utilizes exactly one of the Resources of the specified 'skill';

10.8 Load Size Extensions

10.8.1 load_size_usage_extensions of model Load_Size

FIXED -- a load_size_usage_extension of model Load_Size

Consumes same amount of capacity independent of the number of units of operation_plan.

fixed_quantity -- a Quantity field of model **FIXED**

The fixed quantity of capacity which is consumed.

Default: 0

LINEAR -- a load_size_usage_extension of model Load_Size

Capacity consumption increases as number of units of operation_plan For example, weight

linear_quantity -- a Quantity field of model **LINEAR**

The per_unit capacity that is consumed.

Default: 0

10.9 Flow Extensions

10.9.1 usage_policy_extensions of model Flow

CONSUME_FIXED -- a usage_policy_extension of model Flow

A **CONSUME_FIXED** Flow consumes a fixed 'quantity' of the Item, independent of the number of units of the Operation. For example, you may need to fill the tank with 10 gallons of solvent, no matter how many boards will be dipped.

For compatibility with older systems that do not allow multiple **PRODUCED_FIXED** Flows, a negative quantity indicates that this is produced rather than consumed.

fixed_quantity -- a Quantity field of model **CONSUME_FIXED**

The fixed Quantity of the Item that is produced or consumed by this Operation, independent of the number of units of this Operation performed. An example would be an Operation that requires 100 gallons of plating solution to be put in the tank, independent of the number of boards to be processed by the Operation.

Default: 0

CONSUME_PER -- a usage_policy_extension of model Flow

A **CONSUME_PER** Flow consumes 'quantity_per' of the Item per unit of the Operation.

For compatibility with older systems that do not allow multiple **PRODUCED_PER** Flows, a negative quantity indicates that this is produced rather than consumed.

quantity_per -- a Quantity field of model **CONSUME_PER**

The Quantity of this Item that is consumed per unit of the Operation.

Default: 1

PRODUCED_FIXED -- a usage_policy_extension of model Flow

A **PRODUCED_FIXED** Flow produces 'quantity' of the Item, independent of the number of units of the Operation being performed. For example, 10 gallons of waste solvent may be used up and "produced", no matter how many boards were dipped in the solvent.

<i>Extensions</i>	<i>PRODUCE_PER Extension</i>
-------------------	------------------------------

fixed_quantity -- *a Quantity field of model PRODUCE_FIXED*
The fixed Quantity of the Item that is produced by this Operation, independent of the number of units of this Operation performed. An example would be an Operation that requires 100 gallons of plating solution to be put in the tank, independent of the number of boards to be processed by the Operation.
Default: 0

PRODUCE_PER -- *a usage_policy extension of model Flow*

A **PRODUCE_PER** Flow produces quantity_per of the Item per unit of the Operation.

quantity_per -- *a Quantity field of model PRODUCE_PER*
The Quantity of this Item that is produced per unit of the Operation.
Default: 1

PRODUCE_YIELD -- *a usage_policy extension of model Flow*

The **PRODUCE_YIELD** Flow produces quantity_per of the Item per unit of the Operation, less lossage due to less than 100% yield. Yield is specified as a fixed average value, 'yield' for long-term planning, and a typical worst-case value, 'yield_near', for near-term planning (as specified by near_time). For yield values that can vary over time, see the **PRODUCE_YIELD_CALENDAR** usage_policy extension.

quantity_per -- *a Quantity field of model PRODUCE_YIELD*
The Quantity of this Item that is produced or consumed per unit of the Operation.
Default: 1

yield -- *a Percentage field of model PRODUCE_YIELD*
The expected yield of this Item from this Operation. This should be in the range 0% < yield <= 100%.
Default: 100%

yield_near -- *a Percentage field of model PRODUCE_YIELD*
The worst-case yield that should be used in near-term calculations. This should be in the range 0% < yield_near <= 100%.
Default: 100%

near_time -- *a Time field of model PRODUCE_YIELD*
The time fence defining "near-term" for use of yield_near rather than yield. The purpose is to plan for worst-case yields in the near term, since there is no time to recover. Note that if you planned for worst-case yields over the entire horizon, you may significantly overplan.

<i>Extensions</i>	<i>PRODUCE_YIELD_CALENDAR Extension</i>
-------------------	---

Default: 0

PRODUCE_YIELD_CALENDAR -- *a usage_policy extension of model Flow*

The **PRODUCE_YIELD_CALENDAR** Flow produces quantity_per of the Item per unit of the Operation, less lossage due to less than 100% yield.

The yield is specified much like **PRODUCE_YIELD**: there is an average yield value, 'yield', for long-term planning, and a typical worst-case value, 'yield_near', for near-term planning (as specified by near_time). However, there is also a 'calendar' which specifies multipliers for those yield numbers. The yield changes instantaneously at each Calendar entry. (For linear ramping between entries, see **PRODUCE_YIELD_RAMP_CALENDAR**.)

This is commonly used to reflect process or machine changes that improve yields.

The **PRODUCE_YIELD_CALENDAR** model has fields that references these models:
Calendar.

quantity_per -- *a Quantity field of model PRODUCE_YIELD_CALENDAR*
The Quantity of this Item that is produced or consumed per unit of the Operation.
Default: 1

yield -- *a Percentage field of model PRODUCE_YIELD_CALENDAR*
The base expected yield of this Item from this Operation. This will be multiplied by the appropriate value from the "calendar".
Default: 100%

yield_near -- *a Percentage field of model PRODUCE_YIELD_CALENDAR*
The base worst-case yield that should be used in near-term calculations. This will be multiplied by the appropriate value from the "calendar".
Default: 100%

near_time -- *a Time field of model PRODUCE_YIELD_CALENDAR*
The time fence defining "near-term" for use of yield_near rather than yield. The purpose is to plan for worst-case yields in the near term, since there is no time to recover. Note that if you planned for worst-case yields over the entire horizon, you may significantly overplan.
Default: 0

<i>Extensions</i>	<i>PRODUCE_YIELD_CALENDAR Extension</i>
-------------------	---

calendar - - *a Calendar /field of model* **PRODUCE_YIELD_CALENDAR**
Calendar of factors that are multiplied with the base_yields to give the yield to use in
planning.
Default: [unspecified]

<i>Extensions</i>	<i>Operation_Problem_Detector Extensions</i>
-------------------	--

10.10 Operation_Problem_Detector Extensions
10.10.1 detector extensions of model *Operation_Problem_Detector*

10.11 Operation_Plan Extensions

10.11.1 motive extensions of model Operation_Plan

PRODUCE -- a motive extension of model Operation_Plan

A PRODUCE motive Operation_Plan is performed in order to produce Items with the specified into *buffer_plan* a Quantity between 'min' and 'max' into *buffer_plan* within the due Date_Range (neither before, nor after). The configuration specifies any details about the Item to be produced, according to the Item's spec' fields.

The PRODUCE model has fields that references these models :

Buffer_Plan, *Configuration*.

buffer_plan -- a Buffer_Plan field of model PRODUCE

The Buffer_Plan into which this Operation_Plan should produce the Items. Note that this is the Buffer_Plan that planned this Operation_Plan.

Properties: Export-Only Field

configuration -- a Configuration field of model PRODUCE

The desired characteristics of the Items that this Operation_Plan should produce into *buffer_plan*.

Default: [unspecified]

Properties: Export-Only Field

CONSUME -- a motive extension of model Operation_Plan

A CONSUME motive Operation_Plan is performed in order to consume a Quantity between 'min' and 'max' from *buffer_plan* within the due Date_Range (neither before, nor after). The configuration specifies any details about the Item to be consumed, according to the Item's spec' fields.

The CONSUME model has fields that references these models :

Buffer_Plan, *Configuration*.

buffer_plan -- a Buffer_Plan field of model CONSUME

The Buffer_Plan from which this Operation_Plan should consume the Items. Note that this is the Buffer_Plan that planned this Operation_Plan.

Properties: Export-Only Field

configuration -- a Configuration field of model CONSUME
The desired characteristics of the Items that this Operation_Plan should consume from *buffer_plan*.

Default: [unspecified]

Properties: Export-Only Field

DELIVER -- a motive extension of model Operation_Plan

A DELIVER motive Operation_Plan is performed in order to deliver the a Quantity between 'min' and 'max' from a source Buffer to a specified Location (not a Buffer). The configuration specifies any details about the Item to be consumed, according to the Item's spec' fields.

The DELIVER model has fields that references these models :

Item_Request, *Item_Promise*, *Configuration*.

motive_request -- a Item_Request field of model DELIVER

The Item_Request for which this Operation may be planned to satisfy. This is the same as *motive_promise.item_request*.

Note that this Operation_Plan may not be trying to satisfy this *motive_request*: if planned_for_promise is 'true', then this Operation_Plan is planned to satisfy the

motive_promise rather than this *motive_request*.

Properties: Export-Only Field

Properties: Export-Only Field

motive_promise -- a Item_Promise field of model DELIVER

The Item_Promise for which this Operation may be planned to satisfy. This is the same as *motive_request.item_promise*.

Note that this Operation_Plan may not be trying to satisfy this *motive_promise*: if planned_for_promise is 'false', then this Operation_Plan is planned to satisfy the *motive_request* rather than this *motive_promise*.

Properties: Export-Only Field

planned_for_promise -- a Logical field of model DELIVER

If 'true', then this Operation_Plan has been planned to satisfy the *motive_promise*; otherwise, it is planned to satisfy the *motive_request*. Setting this will cause this Operation_Plan to be replanned accordingly. This can also be set by functions such as *plan_to_satisfy* on the *motive_request* or the *motive_promise*.

Note that the Problems that are detected depend upon how this is set. For example, PROMISE_PLANNED_LATE will only be detected if this is "true". Conversely, REQUEST_PLANNED_LATE will only be detected if this is "false".

Default: false
Properties: Export-Only Field

Item -- a Item field of model DELIVER

The Item which should be delivered to the customer of the Item_Request.
Properties: Export-Only Field

configuration -- a Configuration field of model DELIVER

The desired characteristics of the Items that this Operation_Plan should deliver to the Destination;

Default: [unspecified]

Properties: Export-Only Field

RECEIVE -- a motive extension of model Operation_Plan

A RECEIVE motive Operation_Plan is performed in order to receive the offered quantity from an unsolicited promise. The item specifies which buffer to place the quantity in.

The RECEIVE model has fields that references these models:

Item.

Item -- a Item field of model RECEIVE

The Item which is being received
Properties: Export-Only Field

10.11.2 process extensions of model Operation_Plan

ALTERNATES_PRIMARY -- a process extension of model Operation_Plan

An ALTERNATES_PRIMARY Operation executes one of the sub_operations. It defaults to using the primary Operation, the one with the largest 'percentage'. When it needs to offload, it selects an alternate probabilistically, selecting proportional to 'percentage'.

The difference between ALTERNATES_PRIMARY and ALTERNATES_PROPORTIONAL is that ALTERNATES_PRIMARY always selects the one with the largest 'percentage' initially, but ALTERNATES_PROPORTIONAL makes the initial selection probabilistically also.

move_to_alternate (Operation_Plan, Operation, Quantity), move_to_alternate (Operation_Plan, Operation), move_to_alternate (Operation_Plan), move_to_alternate -- a Void field of model ALTERNATES_PRIMARY

Causes the specified sub_operation_plan to change to the specified alternate (if none specified, then it chooses for you). If no arguments are given, it attempts to alternate itself (ie find the alternate whom it is underneath, and call move_to_alternate on it).

Properties: command=True Export-Only Field

ALTERNATES_PROPORTIONAL -- a process extension of model Operation_Plan

An ALTERNATES_PROPORTIONAL Operation executes one of the sub_operations. It chooses which alternate to use probabilistically, weighed by the Alternate_Operation's 'percentage'.

move_to_alternate (Operation_Plan, Operation, Quantity), move_to_alternate (Operation_Plan, Operation), move_to_alternate (Operation_Plan), move_to_alternate -- a Void field of model ALTERNATES_PROPORTIONAL

Causes the specified sub_operation_plan to change to the specified alternate (if none specified, then it chooses for you). If no arguments are given, it attempts to alternate itself (ie find the alternate whom it is underneath, and call move_to_alternate on it).

Properties: command=True Export-Only Field

EFFECTIVE_CALENDAR -- a process extension of model Operation_Plan

An EFFECTIVE_CALENDAR Operation executes one of the sub_operations. Selection of the sub operation will be done based on the effectivity specified by the calendar for that sub operation. Each sub operation can specify their own effectivity calendar. If more than one sub operation is effective on a given date, probabilistic selection will be done based on the 'percentage' value.

move_to_alternate (Operation_Plan, Operation, Quantity), move_to_alternate (Operation_Plan, Operation), move_to_alternate (Operation_Plan), move_to_alternate -- a Void field of model EFFECTIVE_CALENDAR

Causes the specified sub_operation_plan to change to the specified alternate (if none specified, then it chooses for you). If no arguments are given, it attempts to alternate itself (ie find the alternate whom it is underneath, and call move_to_alternate on it).

Properties: command=True Export-Only Field

DELAY_ONLY_FIXED -- a process extension of model Operation_Plan

Extensions	DELAY_ONLY_BASIC Extension
------------	----------------------------

A DELAY_ONLY_FIXED Operation loads no Resource. It simply runs for a fixed amount of time, independent of the quantity being processed. Thus, this can be used to insert a fixed delay in the processing.

DELAY_ONLY_BASIC -- a process extension of model Operation_Plan

A DELAY_ONLY_BASIC Operation loads no Resource. It simply runs for a time equal to `Fixed_time + units*time_per`. Thus, this can be used to insert a per-unit delay in the processing.

BASIC_CALENDARS -- a process extension of model Operation_Plan

A BASIC_CALENDARS Operation loads the Resources for a time equal to `time + units*time_per/resource.efficiency`, where both the time per unit of the operation (`time_per`) and the additional operation time (`time`) are specified using calendars.

FIXED_TIME -- a process extension of model Operation_Plan

A FIXED_TIME Operation loads a Resource for a fixed amount of Time, independent of the Quantity being processed.

TIME_MULTIPLE -- a process extension of model Operation_Plan

A TIME_MULTIPLE Operation loads a Resource for a Time that increases in discrete multiples of `base_time`; based on the number of units as a multiple of `base_units`. More mathematically, the time is `round_up(units / base_units) * base_time`.

BASIC -- a process extension of model Operation_Plan

A BASIC Operation loads the Resources for a time equal to `time + units*time_per/resource.efficiency`.

BASIC_DELAYED -- a process extension of model Operation_Plan

A BASIC_DELAYED Operation loads the Resources for a time equal to `time + units*time_per/resource.efficiency`. In addition, the BASIC_DELAYED Operation total time includes a fixed `warm_up_delay` that occurs prior to loading the Resources and a fixed `cool_down_delay` that occurs after loading the Resources.

Note that delay Times are not affected by the efficiency of the Resources.

REQUEST_FIXED -- a process extension of model Operation_Plan

Extensions	REQUEST_FIXED_WITH_ANALYSIS Extension
------------	---------------------------------------

A REQUEST_FIXED Operation places a Request for an Item Item' at another Site (supplier). The Operation requires time from completion of delivery of the Item' by the supplier' to completion of this Operation (when the output Flows will produce to those Buffers).

The REQUEST_FIXED model has fields that references these models :
Item_Request.

Item_request -- a Item_Request field of model REQUEST_FIXED

The Item_Request which is generated by this Operation_Plan.
Properties: Export-Only Field

REQUEST_FIXED_WITH_ANALYSIS -- a process extension of model Operation_Plan

A REQUEST_FIXED_WITH_ANALYSIS Operation places a Request for an Item item' at another Site (supplier). The Operation requires time from completion of delivery of the item' by the supplier' to completion of this Operation (when the output Flows will supply to those Buffers).

The REQUEST_FIXED_WITH_ANALYSIS model has fields that references these models :
Item_Request.

Item_request -- a Item_Request field of model

REQUEST_FIXED_WITH_ANALYSIS

The Item_Request which is generated by this Operation_Plan.

Properties: Export-Only Field

ROUTING -- a process extension of model Operation_Plan

A ROUTING Operation executes the sub_operations' in order, with no overlap. The sub_operations and their ordering are defined by the routing_operations;

It can add Loads which consume Resources for the duration of the sub_operations, or it can add Flows which consume Items at the beginning of the sub_operations' or produce Items at the end.

Note that a ROUTING Operation does not have any 'sid_time' of its own (its always "0"). If Resources are loaded by this ROUTING Operation, then the efficiency of the Resource affects all sub_operations, but does not affect the time between sub_operations. For example, if a Resource used by this ROUTING Operation drops to 50%, then the maximum efficiency of any sub Operation will be 50% (if it was at 100%, the load time required will double).

10.12 Resource Extensions

10.12.1 efficiency extensions of model Resource

FIXED -- a efficiency extension of model Resource

A FIXED efficiency Resource has a fixed efficiency level at all times. The default value is the very common case of 100%.

fixed_efficiency -- a Percentage field of model FIXED

The efficiency level (how fast it is processing).

Default: 100%

CALENDAR -- a efficiency extension of model Resource

A CALENDAR efficiency Resource has efficiency that will vary over time as specified in a Calendar. This may be used to model a shift calendar, with potentially different efficiency in different shifts and downtimes. It may also be used to represent any efficiency with cyclic behavior relative to the calendar. Or it may be used to model in detail a ramp up, ramp down, or engineering change to become effective.

Further, the options for increasing efficiency by incurring additional cost can be modeled. Calendar_Entry's with a non-zero 'charge' are treated as options for increasing efficiency when needed.

The CALENDAR model has fields that references these models :

Calendar.

efficiency_calendar -- a Calendar field of model CALENDAR

A Calendar with PERCENTAGE entries, where the Percentage is the efficiency level at those dates. Entries with a non-zero 'charge' are treated as options for increasing efficiency when needed.

Default: [unspecified]

10.12.2 variability extensions of model Resource

ZERO -- a variability extension of model Resource

A ZERO variability Resource does not have significant enough variability to need planning compensation or padding.

FIXED -- a variability extension of model Resource

<i>Extensions</i>	<i>maintenance extensions of model Resource</i>
-------------------	---

A **FIXED** variability Resource has variability that is compensated for by padding the arrival of upstream Operations and the departure to downstream Operations by fixed Times.

The 'downstream_pad' reduces the effects that the variability of this Resource has upon the downstream Resources. So, if this machine goes down for 1 hour every 10 days, you may want to set 'downstream_pad' to 1 hour. In that way, all downstream Operations and Buffers are expecting to receive the material an hour late -- and thus, their schedules are not disturbed when the machine goes down for an hour.

The 'upstream_pad' prevents this Resource from starving when it runs faster than expected, or allows it to run things in a different order by providing it a certain level of WIP in front of it. If this machine sometimes gets as much as an hour ahead of schedule, you may want to set 'upstream_pad' to 1 hour. In that way, all upstream Operations and Buffers plan to produce the material 1 hour early, and thus the machine does not starve when it gets an hour ahead.

Note that increasing these pads will increase WIP on the floor. Thus, the pads should generally be kept as small as possible without making the plan too brittle in the face of disturbances. The 'upstream_pad' may not be important on any but the most utilized Resources (the ones where you do not want it to ever starve).

upstream_pad -- *a Time field of model FIXED*

The extra Time by which to pad the arrival from upstream Operations to Operations on this Resource.

Default: 0

downstream_pad -- *a Time field of model FIXED*

The extra Time by which to pad the departure to downstream Operations from Operations on this Resource.

Default: 0

10.12.3 maintenance extensions of model Resource

ZERO -- *a maintenance extension of model Resource*

A ZERO maintenance Resource does not need any maintenance.

10.12.4 size extensions of model Resource

UNLIMITED -- *a size extension of model Resource*

<i>Extensions</i>	<i>FIXED_COUNT Extension</i>
-------------------	------------------------------

An **UNLIMITED** size Resource has no relevant size constraints. Although physically unreasonable, this is very common for planning purposes. Most Operations are designed to fit on the Resources they use, and thus cannot be "too big". All Operations are considered to have size of "1".

FIXED_COUNT -- *a size extension of model Resource*

A **FIXED_COUNT** size Resource has a single fixed size limit at all times. The size of an Operation_Plan is "1". Thus, the 'max_operation_count' is the maximum number of Operation_Plans that can run on the Resource at once. This is commonly used to model several identical Resources as one Resource. The 'max_operation_count' is effectively the number of identical Resources that are available.

The default value is the common case of "oo" (infinite) which means there are no size limits.

max_operation_count -- *a Integer field of model FIXED_COUNT*

The maximum number of Operation_Plans that can use this Resource at once.

Default: oo (infinite, unlimited, unless)

FIXED_QUANTITY -- *a size extension of model Resource*

A **FIXED_QUANTITY** size Resource has a single fixed size limit at all times. The size of an Operation is measured by converting the units of the Operation_Plan to the unit of Measure of the 'max_size'. Thus, if the 'max_size' is "500 kg", the size of an Operation_Plan will be the units of the Operation_Plan converted to "kg". If the 'max_size' is unitless, then the number of units of the Operation_Plan is used.

The default value is the common case of "oo" (infinite) which means there are no size limits.

max_size -- *a Quantity field of model FIXED_QUANTITY*

The maximum total size of Operation_Plans that can run on this Resource at once.

Default: oo (infinite, unlimited, unless)

CALENDAR_COUNT -- *a size extension of model Resource*

A **CALENDAR_COUNT** size Resource can handle a maximum number of Operation_Plans at once that varies over time as specified in a Calendar. Calendar_Entry's with a non-zero 'change' are treated as options for increasing size when needed.

Extensions	MULTI_DIMENSION Extension
------------	---------------------------

The size of an Operation_Plan is "1". Thus, the Number in the Calendar is the maximum number of Operation_Plan that can run on the Resource at once. This is commonly used to model several identical Resources as one Resource. The Number in each Calendar_Entry is effectively the number of identical Resources that are available.

The CALENDAR_COUNT model has fields that references these models :

Calendar:

size_calendar -- a Calendar field of model CALENDAR_COUNT

A Calendar with NUMBER entries, where the Quantity is the maximum number of Operation_Plan.

Default: [unspecified]

MULTI_DIMENSION -- a size extension of model Resource

A MULTI_DIMENSION size resource can model size limits in multiple dimensions. Each dimension can be declared with a min_size and a max_size limit.

'aggregate_count' represents the total number of resources available. Each of these 'aggregate_count' resources are loaded the same way using the load_policy. For example, the EXCLUSIVE_USE load_policy will load 'aggregate_count' resources, each of which is of type EXCLUSIVE_USE. The SIMPLE_CONSOLIDATION load_policy will load 'aggregate_count' resources, each of which has consolidation restriction specified by 'dimensions'.

Each resource has size restrictions as specified in 'dimensions'. The loads on this resource should have 'load_sizes' defined in all the 'dimensions' specified here. The load_policy decided how to use the various dimensions. For example EXCLUSIVE_USE, SHARED_USE and SIMPLE_CONSOLIDATION load_policies consider the dimensions' orthogonally. Violation in any one dimension flags problems. Future load_policies can consider violations in only one dimension or combine them arbitrarily.

The MULTI_DIMENSION model has these submodels :

Size_Dimension.

The MULTI_DIMENSION model has fields that references these models :

Size_Dimension.

Extensions	load_policy extensions of model Resource
------------	--

dimensions -- a list of Size_Dimension submodels of model MULTI_DIMENSION
The various dimensions of the size of each one of the aggregate resource. The dimensions need not be orthogonal.

10.12.5 load_policy extensions of model Resource

SIMPLE_CONSOLIDATION -- a load_policy extension of model Resource

An SIMPLE_CONSOLIDATION Resource lets users consolidate the load_plans on this resource. No loading Problems will be detected or resolved (such as two Operations planned at once).

consolidation_fence -- a Horizon_Date field of model

SIMPLE_CONSOLIDATION

The fence after which no consolidation problems are detected

Default: 'infinite future'

INFINITE_USE -- a load_policy extension of model Resource

An INFINITE_USE Resource is considered to have "infinite capacity" in the traditional sense. That is, each Operation is loaded appropriately (considering its capacity), but without considering any other Operations. No loading Problems will be detected or resolved (such as two Operations planned at once). But you can still view the planned load versus the actual capacity available.

EXCLUSIVE_USE -- a load_policy extension of model Resource

An EXCLUSIVE_USE Resource can handle only one load at any given time. All overlapping loads are identified as Problems, size of the resource is ignored in planning the Resource. This is done deliberately to have fast run times for this most common model. It is an error to have a size other than UNLIMITED (the default) for this extension.

SHARED_USE -- a load_policy extension of model Resource

A SHARED_USE Resource can handle multiple Loads at the same time. The Loads are allowed to overlap arbitrarily, as long as they do not exceed the size limits at that time as specified by the size extension.

Extensions

SHARED_USE Extension

For example, if there is a **FIXED_QUANTITY** size limit of "2 tons", then up to 2 tons of Loads may be simultaneously processed. If there is a **FIXED_COUNT** size limit of "5", then up to 5 Loads will be allowed at once. Note that if there is a **FIXED_COUNT** size limit of "1", then this is equivalent to **EXCLUSIVE_USE** (just one Load at a time).

All Loads are considered compatible. In contrast, a **SHARED_USE_SETUP** Resource can only run together Operations with identical setups. A common example would be 10 identical Resources that are preferably modeled as one Resource that can run 10 things at once.

The **SHARED_USE** model has fields that references these models :

Horizon.

buckets - - *a Horizon field of model SHARED_USE*

The Horizon that defines the 'dates' of the buckets used by this policy after the **horizon_fence**' date.

Default: none

bucket_fence - - *a Horizon_Date field of model SHARED_USE*

The horizon-relative Date beyond which this **load_policy** switches to coarser planning estimates. After the fence Date, this **load_policy** will do bucket based problem detection and resolution. This date is relative to the plan.start

Default: none

min_load - - *a Percentage field of model SHARED_USE*

The minimum load on this resource, expressed as a percentage of the size. The resource has to have at least **min_load** from Plan.start to **min_load_fence**. For example, if the **min_load** is 80%, then this resource has to be utilized by at least 80% of its capacity from Plan.start to **min_load_fence**. If the utilization goes below 80%, we flag an **UNDERLOAD** problem from Plan.start till **min_load_fence**. The valid values of the fields are from 0% to 100%. The default value of **min_load** is 0%, meaning there is no restriction.

Default: 0%

min_load_fence - - *a Horizon_Date field of model SHARED_USE*

The relative date from Plan.start till which we do not want this resource to be underutilized.

Default: none

Extensions

SHARED_USE Extension

max_bucket_load - - *a Percentage field of model SHARED_USE*

The resource cannot be over-utilized by more than **max_bucket_load** in any bucket. This is expressed as a percentage of the size in actual hours of the resource in this bucket. (resource_Plan.available_time). For example, if the **max_bucket_load** is 80%, then this resource cannot be utilized by more than 80% of its capacity in any bucket. If the total utilization goes above 80% of the total capacity, a **BUCKET_OVERSIZE** problem will be flagged in that bucket. The default value of **max_bucket_load** is 100%, meaning the resource cannot be loaded more than 100% of its capacity.

Default: 100%

upstream_bucket_pad - - *a Time field of model SHARED_USE*

The extra Time by which to pad the arrival from upstream Operations to Operations on this Resource. This pad is in addition to any upstream pad imposed by the variability extension. This is the upstream pad needed to have a stable plan when switching from bucketized planning to detailed planning. This will be added to the operation_plan only if it starts loading the resource at or after the **bucket_fence**.

Default: 0

downstream_bucket_pad - - *a Time field of model SHARED_USE*

The extra Time by which to pad the departure to downstream Operations from Operations on this Resource. This pad is in addition to any downstream pad imposed by the variability extension. This is the downstream pad needed to have a stable plan when switching from bucketized planning to detailed planning. This will be added to the operation_plan only if it ends loading the resource after the **bucket_fence**.

Default: 0

<i>Extensions</i>	<i>Resource_Skill_Extensions</i>
-------------------	----------------------------------

10.13 Resource_Skill_Extensions

10.13.1 efficiency_extensions of model Resource_Skill

FIXED -- a efficiency_extension of model Resource_Skill

A FIXED efficiency Resource_Skill has a fixed efficiency level at all times. The default value is the very common case of 100%.

fixed_efficiency -- a Percentage field of model FIXED
The efficiency level of the resource in performing this skill
Default: 100%

CALENDAR -- a efficiency_extension of model Resource_Skill

A CALENDAR efficiency Resource_Skill has efficiency that will vary over time as specified in a Calendar.

Further, the options for increasing efficiency by incurring additional cost can be modeled. Calendar_Entry's with a non-zero 'charge' are treated as options for increasing efficiency when needed. Uses the Default_Efficiency_Calendar, if efficiency_calendar is not set.

The CALENDAR model has fields that references these models :
Calendar.

efficiency_calendar -- a Calendar field of model CALENDAR
A Calendar with PERCENTAGE 'entries', where the Percentage is the efficiency level at those dates. Entries with a non-zero 'charge' are treated as options for increasing efficiency when needed.
Default: [unspecified]

<i>Extensions</i>	<i>Resource_Problem_Detector_Extensions</i>
-------------------	---

10.14 Resource_Problem_Detector_Extensions

10.14.1 detector_extensions of model Resource_Problem_Detector

<i>Extensions</i>	<i>Resource_Plan Extensions</i>
-------------------	---------------------------------

10.15 Resource_Plan Extensions

10.15.1 efficiency extensions of model Resource_Plan

10.15.2 load_policy extensions of model Resource_Plan

SIMPLE_CONSOLIDATION -- a load_policy extension of model Resource_Plan

An SIMPLE_CONSOLIDATION Resource lets users consolidate the load_plans on this resource. No load Problems will be detected or resolved (such as two Operations planned at once).

The SIMPLE_CONSOLIDATION model has these submodels :
Consolidation.

The SIMPLE_CONSOLIDATION model has fields that references these models :
Consolidation.

consolidations - - a list of Consolidation submodels of model
SIMPLE_CONSOLIDATION
'consolidations' are created to group load_plans together.

unconsolidated_operation_plans - - a List(Operation_Plan) field of model
SIMPLE_CONSOLIDATION

The operation_plans that are planned on this resource but do not belong to any consolidation

Properties: Export-Only Field

INFINITE_USE -- a load_policy extension of model Resource_Plan

No loading Problems will be detected or resolved (such as two Operations planned at once). But you can still view the planned load versus the actual capacity available.

EXCLUSIVE_USE -- a load_policy extension of model Resource_Plan

All overlapping loads are identified as Problems. size of the resource is ignored in planning the Resource. This is done deliberately to have fast run times for this most common model.

SHARED_USE -- a load_policy extension of model Resource_Plan

<i>Extensions</i>	<i>size extensions of model Resource_Plan</i>
-------------------	---

A SHARED_USE Resource can handle multiple Loads at the same time. The Loads are allowed to overlap arbitrarily, as long as they do not exceed the size limits at that time as specified by the 'size' extension.

For example, if there is a FIXED_QUANTITY size limit of "2 tons", then up to 2 tons of Loads may be simultaneously processed. If there is a FIXED_COUNT size limit of "5", then up to 5 Loads will be allowed at once. Note that if there is a FIXED_COUNT size limit of "1", then this is equivalent to EXCLUSIVE_USE (just one Load at a time).

All Loads are considered compatible. In contrast, a SHARED_USE_SETUP Resource can only run together Operations with identical setups. A common example would be 10 identical Resources that are preferably modeled as one Resource that can run 10 things at once.

10.15.3 size extensions of model Resource_Plan

10.15.4 maintenance extensions of model Resource_Plan

Extensions	Buffer Extensions
------------	-------------------

10.16 Buffer Extensions

10.16.1 Flow_policy extensions of model Buffer

BUCKETED_NESTED_SORT -- a flow_policy extension of model Buffer

A BUCKETED_NESTED_SORT Buffer manages the Flow_Plan in buckets of time, as specified by the horizon.

Within each bucket, the consuming Flow_Plans are sorted by doing a nested sort on one or more criteria: the primary sort key (the highest ranking criteria) is used first, the second criteria is used to break ties in the first sort, and so on. Within each bucket, the producing Flow_Plan generated by this Buffer can be broken out according to a subset of the criteria that are being used.

A target ending_on_hand will be maintained at the end of the bucket, as computed from the sum of a fixed Quantity and a percentage of the flow in the next bucket.

The BUCKETED_NESTED_SORT flow_policy should preferably be used on end Item Buffers (buffers whose consuming operation plans have a DELIVERY motive). Many of the sort criteria function well only if this is an end Item Buffer, though (if the delivery Operations for the Requests consume directly from this Buffer). In the future we will provide mechanisms for extending the visibility of Request properties beyond the end Item Buffers. See the Flow_Criterion extensions for more information on their behavior.

The BUCKETED_NESTED_SORT model has these submodels:

Flow_Criterion.

The BUCKETED_NESTED_SORT model has fields that references these models:

Operation, Horizon, Product_Root, Flow_Criterion.

producing_operation -- a Operation field of model

BUCKETED_NESTED_SORT

The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field_Error for this field to be [unspecified], the default, if any Buffer_Plan needs to issue a producing Operation_Plan.

Default: [unspecified]

Extensions	BUCKETED_NESTED_SORT Extension
------------	--------------------------------

horizon -- a Horizon field of model BUCKETED_NESTED_SORT
The Horizon that defines the 'dates' of the buckets used by this policy.

Default: [unspecified]

fixed_ending_on_hand -- a Quantity field of model

BUCKETED_NESTED_SORT

The on_hand to be present at the end of each bucket should be the sum of this 'fixed_ending_on_hand' and the per_next_ending_on_hand multiplied by the consuming_flow from the next bucket.

The net target value for each bucket in a particular Plan is available in the 'ending_on_hand' field of the priority_buckets of the Buffer_Plan. That computed value is also settable, allowing the user to override the target 'ending_on_hand' for a bucket without modifying these Buffer fields.

This Quantity will be converted into the unit of this Buffer.

Default: 0

per_next_ending_on_hand -- a Percentage field of model

BUCKETED_NESTED_SORT

The on_hand to be present at the end of each bucket should be the sum of the 'fixed_ending_on_hand' and this per_next_ending_on_hand multiplied by the consuming_flow from the next bucket. If there is no next bucket, then this bucket is used.

The net target value for each bucket in a particular Plan is available in the 'ending_on_hand' field of the priority_buckets of the Buffer_Plan. That computed value is also settable, allowing the user to override the target 'ending_on_hand' for a bucket without modifying these Buffer fields.

Default: 0%

default_product_root -- a Product_Root field of model

BUCKETED_NESTED_SORT

When this field is specified, EXCESS_ON_HAND problems are resolved by creating forecast promises to the 'default_product_root'. In this way the Bucketed_Nested_Sort policy ensures that all excess material is allocated to the seller hierarchy instead of sitting in the buffer. Note that for the default value of this field, EXCESS_ON_HAND problems are not resolved.

Default: [unspecified]

do not move out forecasts -- a logical field of model
BUCKETED_NESTED_SORT
Defaults to true, when resolving Negative. On_Hand problems, don't move out forecast consumers. They will fall at the end of the current bucket and should not be moved into the next bucket, rather they should be shorted.
Default: true

split_multiple -- a Quantity field of model **BUCKETED_NESTED_SORT**
Splits made to Operation_Plan by a **BUCKETED_NESTED_SORT** Buffer will always be in an amount that is a multiple of this value. This Quantity will be converted into the unit of this Buffer.
Default: 1
Properties: obsolete=True

criteria -- a list of Flow_Criterion submodels of model
BUCKETED_NESTED_SORT

PRODUCING_FLOW_CALENDAR -- a flow_policy extension of model Buffer

A **PRODUCING_FLOW_CALENDAR** Buffer is replenished on a set schedule specified by the 'calendar'. This Calendar based replenishments are only observed during the period specified via 'fence'. This flow_policy will behave like **MULTIPLE** or **BASIC**(if the 'after_fence_quantity_range' at and after the 'fence'. Buffer can plan a producing operation on any date as needed by the consuming **Operation_Plan**.

In conjunction with Calendar based replenishments, this flow_policy will allow users to specify lot sizing restrictions in terms of min, max and multiple. The quantity of each replenishment order is in multiples of 'multiple_quantity' and is bounded by 'quantity_range'.

Lot sizing restrictions can be relaxed by specifying different
'after_fence_multiple_quantity' and 'after_fence_quantity_range' at and beyond the 'fence'. This Horizon_Date 'fence' is relative to plan start and will move with it.

This Buffer will also maintain material in quantity greater than or equal to 'min_on_hand' and less than or equal to 'excess_on_hand' in an any given period. Both of these quantities can be date effective and specified via set of Date_Ranges. By setting 'excess_on_hand' larger than zero, planning instability is reduced, and planning speed is increased. However, keeping it smaller minimizes excess inventory.

User can also specify date effective 'min_time' which gives a fixed time of consumption that should always be covered (for example, setting 'min_time' to '3 weeks' will keep enough stock to cover the next 3 weeks of out flows).

The **PRODUCING_FLOW_CALENDAR** model has fields that references these models:
Calendar, Operation.

calendar -- a Calendar field of model **PRODUCING_FLOW_CALENDAR**
A Calendar of entries that define the preset schedule of replenishments. Each **Calendar_Entry** will specify number of replenishments allowed on a given date. For example, when set to 1, it will only plan one producing operation on each effective schedule date. The default value for calendar entries is 0, see the **Calendar_Entry** value **NUMBER** extension.

Size of each replenishment is computed to maintain at least the 'min_on_hand' until the next replenishment observing the lot sizing restrictions.
Default: [unspecified]

quantity_range -- a Quantity_Range field of model
PRODUCING_FLOW_CALENDAR

The Quantity_Range between which producing operations of this buffer are constrained to lie.

Note that the smallest legal size of a producing operation can be more than the min of the quantity_range, since it has to be a multiple of multiple_quantity. Similarly, the largest legal size can be smaller than the max of the quantity_range.
Default: [1, oo]

multiple_quantity -- a Quantity field of model
PRODUCING_FLOW_CALENDAR

The Quantity of which the Quantity of the Item to be produced by each producing_operation that is issued by this **PRODUCING_FLOW_CALENDAR** Buffer is a multiple of.

This Quantity is converted to the unit of this Buffer. If multiple_quantity is set to "0", it implies that the quantities of producing operations can be any number (integer or non-integer) in the range of "quantity_range". The default is "0".
Default: 0

fence - - *a* **Horizon**, *Date field of model* **PRODUCING_FLOW_CALENDAR**
 The horizon-relative Date beyond which this flow_policy switches to coarser planning estimates. At and after the fence Date producing_operation issued by this **PRODUCING_FLOW_CALENDAR** buffer will use **after_fence_quantity_range** instead of **quantity_range** and **after_fence_multiple_quantity** instead of **multiple_quantity** as lot sizing restrictions.

Also, at and after this 'fence', replenishment calendar is not effective causing this flow policy to behave like **MULTIPLE**. At and after the 'fence' date, it can create producing **Operation_Plan** as needed.

This fence is relative to Plan.current.

Default: 00

after_fence_quantity_range - - *a* **Quantity**, *Range field of model* **PRODUCING_FLOW_CALENDAR**

The range between which producing operations of the buffer are constrained to lie at and after fence.

Note that the smallest legal size of a supplying operation can be more than the min of the quantity_range, since it has to be a multiple of **multiple_quantity**. Similarly, the largest legal size can be smaller than the max of the quantity_range.
 Default: [1, 00]

after_fence_multiple_quantity - - *a* **Quantity**, *field of model* **PRODUCING_FLOW_CALENDAR**

The Quantity of the item to be produced by each producing_operation that is issued by this Buffer is a multiple of at and after the fence. This Quantity is converted to the unit of this Buffer. An **after_fence_multiple_quantity** of "0" implies that the quantities of producing operations can be any number in the range of "quantity_range". The default is "0".
 Default: 0

default_min_on_hand - - *a* **Quantity**, *field of model* **PRODUCING_FLOW_CALENDAR**

The default minimum on_hand balance that should be present in this Buffer. This will be effective during the periods when there are no Calendar_Entry's specified in the **min_on_hand** Calendar.

If the **min_on_hand** Calendar is [unspecified], the **default_min_on_hand** will be effective through out the planning horizon.

If the stocking_policy is **CALENDAR** then **default_min_on_hand** will be automatically computed.

Default: 0

min_on_hand_calendar - - *a* **Calendar**, *field of model* **PRODUCING_FLOW_CALENDAR**

A Calendar of entries that define minimum on_hand balance that should be maintained in this Buffer at those dates. These Quantity's are converted to the unit of this Buffer.

If the stocking_policy is **CALENDAR** then **min_on_hand_calendar** will be automatically computed.

Default: [unspecified]

default_excess_on_hand - - *a* **Quantity**, *field of model* **PRODUCING_FLOW_CALENDAR**

Up to this Quantity more than the minimum required is allowed in this Buffer. This is the default excess on_hand Quantity effective during the periods when there are no Calendar_Entry's specified in the **excess_on_hand** Calendar.

If the **excess_on_hand** Calendar is [unspecified], the **default_excess_on_hand** will be effective through out the planning horizon.

Default: 0

excess_on_hand_calendar - - *a* **Calendar**, *field of model* **PRODUCING_FLOW_CALENDAR**

A Calendar of entries that define Quantity more than the minimum required is allowed in this Buffer. These Quantity's are converted to the unit of this Buffer.

Setting **excess_on_hand** above zero reduces instability, and speeds planning. Setting it smaller decreases excess inventory levels.

Default: [unspecified]

default_min_time - - *a* **Time**, *field of model* **PRODUCING_FLOW_CALENDAR**

This is the default min_time effective during the periods when there are no Calendar_Entry's specified in the **min_time** Calendar.

Items are planned such that they arrive at least 'default_min_time' earlier than otherwise needed. This Buffer will allow specification of date effective min_time which can vary over time.

If the stocking_policy is CALENDAR and if the units_of_safety_stock is either TIME or QUANTITY_TIME then default_min_time will be automatically computed.

min_time is represented internally as seconds so if specified as as days, weeks, years, its will be converted to seconds. This needs to be considered around daylight savings time boundaries.

Default: 0

min_time_calendar -- a Calendar field of model

PRODUCING_FLOW_CALENDAR

A Calendar of entries that define min_time effective during those dates. Items are planned such that they arrive at least min_time earlier than otherwise needed. This Buffer will allow specification of date effective min_time which can vary over time.

This may be used as a safety time -- the next min_time of consumption will be maintained in the Buffer.

The 'min_time' gives a fixed time of consumption that should always be covered (for example, setting 'min_time' to "3 weeks" will keep enough stock to cover the next 3 weeks of our flows).

If the stocking_policy is CALENDAR and if the units_of_safety_stock is either TIME or QUANTITY_TIME then min_time_calendar will be automatically computed.

Default: [unspecified]

end_item -- a Logical field of model PRODUCING_FLOW_CALENDAR

If TRUE, this Buffer will assume that majority of it's consuming Operation_Plans have DELIVER motive and are connected directly to the Item, Requests/Promises. It will take advantage of this proximity to Requests/Promises and resolve problems by selecting Operation_Plans and actions(MOVE_IN, MOVE_OUT, RESIZE_MORE, RESIZE_LESS etc.), which will result into reduced lateness, shortness or in general most positive impact on the Active_Goals.

If FALSE, this Buffer will use internal heuristics to select Operation_Plans and actions(MOVE_IN, MOVE_OUT, RESIZE_MORE, RESIZE_LESS etc.) to solve Problems.

Default: false

rank_delivery_operation_plan_higher -- a Logical field of model

PRODUCING_FLOW_CALENDAR

If TRUE, consuming Operation_Plans motivated directly by Requests and Promises via DELIVER motive are ranked higher than the consuming Operation_Plans motivated by downstream Buffers or Resources.

If FALSE, consuming Operation_Plans motivated directly by Requests and Promises via DELIVER motive are ranked lower. This will be used by problem resolvers to rank candidates while taking Strategy allowed actions such as MOVE_OUT, RESIZE_LESS.

Default: true

producing_operation -- a Operation field of model

PRODUCING_FLOW_CALENDAR

The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field_Error for this field to be [unspecified], the default, if any Buffer_Plan needs to issue a producing Operation_Plan.

Default: [unspecified]

PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK -- a flow_policy extension of model Buffer

Just like Producing Flow Policy, but with user entry points to control resolver behavior

The PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK model has fields that references these models :

Calendar, Operation.

calendar -- a Calendar field of model

PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK

A Calendar of entries that define the preset schedule of replenishments. Each Calendar_Entry will specify number of replenishments allowed on a given date.

Defaulted to 0, it will not plan producing operation on each effective schedule date unless the size is set.

Size of each replenishment is computed to maintain at least the 'min_on_hand' until the next replenishment observing the lot sizing restrictions.

Default: [unspecified]

quantity_range - - *a* Quantity Range *field of model*
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK
 The Quantity_Range between which producing operations of this buffer are constrained to lie.

Note that the smallest legal size of a producing operation can be more than the min of the quantity_range, since it has to be a multiple of multiple_quantity. Similarly, the largest legal size can be smaller than the max of the quantity_range.
 Default: [1, oo]

multiple_quantity - - *a* Quantity *field of model*
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK
 The Quantity of which the Quantity of the Item to be produced by each producing_operation that is issued by this PRODUCING_FLOW_CALENDAR_Buffer is a multiple of.

This Quantity is converted to the unit of this Buffer. The default is "1", which means 1 unit of the Buffer.
 Default: 1

fence - - *a* Horizon Date *field of model*
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK
 The horizon-relative Date beyond which this flow_policy switches to coarser planning estimates. At and after the fence Date producing_operation issued by this PRODUCING_FLOW_CALENDAR buffer will use after_fence_quantity_range instead of quantity_range and after_fence_multiple_quantity instead of multiple_quantity as lot sizing restrictions.

Also, at and after this 'fence', replenishment calendar is not effective causing this flow policy to behave like MULTIPLE. After the Fence date, it can create producing Operation_Plan as needed.

This fence is relative to Plan.current.
 Default: oo

after_fence_quantity_range - - *a* Quantity Range *field of model*
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK
 The range between which producing operations of the buffer are constrained to lie at and after fence.

Note that the smallest legal size of a supplying operation can be more than the min of the quantity_range, since it has to be a multiple of multiple_quantity. Similarly, the largest legal size can be smaller than the max of the quantity_range.
 Default: [1, oo]

after_fence_multiple_quantity - - *a* Quantity *field of model*
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK
 The Quantity of the Item to be produced by each producing_operation that is issued by this Buffer is a multiple of at and after the fence. This Quantity is converted to the unit of this Buffer. The default is "1", which means 1 unit of the Buffer.
 Default: 1

default_min_on_hand - - *a* Quantity *field of model*
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK
 The default minimum on_hand balance that should be present in this Buffer. This will be effective during the periods when there are no Calendar_Entry's specified in the min_on_hand Calendar.

If the 'min_on_hand' Calendar is [unspecified], the 'default_min_on_hand' will be effective through out the planning horizon.
 Default: 0

min_on_hand_calendar - - *a* Calendar *field of model*
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK
 A Calendar of entries that define minimum on_hand balance that should be maintained in this Buffer at those dates. These Quantity's are converted to the unit of this Buffer.
 Default: [unspecified]

default_excess_on_hand - - *a* Quantity *field of model*
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK
 Up to this Quantity more than the minimum required is allowed in this Buffer. This is the default excess on_hand Quantity effective during the periods when there are no Calendar_Entry's specified in the excess_on_hand Calendar.

If the 'excess_on_hand' Calendar is [unspecified], the 'default_excess_on_hand' will be effective through out the planning horizon.
 Default: 0

excess_on_hand_calendar -- *a Calendar field of model*

PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK

A Calendar of entries that define Quantity more than the minimum required is allowed in this Buffer. These Quantity's are converted to the unit of this Buffer.

Setting **excess_on_hand** above zero reduces instability, and speeds planning. Setting it smaller decreases excess inventory levels.

Default: [unspecified]

default_min_time -- *a Time field of model*

PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK

This is the default **min_time** effective during the periods when there are no

Calendar_Entry's specified in the **min_time** Calendar.

Items are planned such that they arrive at least 'default_min_time' earlier than otherwise needed. This Buffer will allow specification of date effective **min_time** which can vary over time.

min_time is represented internally as seconds so if specified as days, weeks, years, it will be converted to seconds. This needs to be considered around daylight savings time boundaries.

Default: 0

min_time_calendar -- *a Calendar field of model*

PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK

A Calendar of entries that define **min_time** effective during those dates. Items are planned such that they arrive at least **min_time** earlier than otherwise needed. This Buffer will allow specification of date effective **min_time** which can vary over time.

This may be used as a safety time -- the next **min_time** of consumption will be maintained in the Buffer.

The **min_time** gives a fixed time of consumption that should always be covered (for example, setting **min_time** to "3 weeks" will keep enough stock to cover the next 3 weeks of out flows).

Default: [unspecified]

end_item -- *a Logical field of model*

PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK

If **TRUE**, this Buffer will assume that majority of it's consuming **Operation_Plan**s have **DELIVER** motive and are connected directly to the **Item_Requests/Promises**. It will take advantage of this proximity to **Requests/Promises** and resolve problems by selecting **Operation_Plan**s and actions(**MOVE_IN**, **MOVE_OUT**, **RESIZE_MORE**, **RESIZE_LESS** etc.), which will result into reduced lateness, shortness or in general most positive impact on the **Active_Goals**.

If **FALSE**, this Buffer will use internal heuristics to select **Operation_Plan**s and actions(**MOVE_IN**, **MOVE_OUT**, **RESIZE_MORE**, **RESIZE_LESS** etc.) to solve Problems.

Default: false

rank_delivery_operation_plan_higher -- *a Logical field of model*

PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK

If **TRUE**, consuming **Operation_Plan**s motivated directly by **Requests** and **Promises** via **DELIVER** motive are ranked higher than the consuming **Operation_Plan**s motivated by downstream Buffers or Resources.

If **FALSE**, consuming **Operation_Plan**s motivated directly by **Requests** and **Promises** via **DELIVER** motive are ranked lower. This will be used by problem solvers to rank candidates while taking Strategy allowed actions such as **MOVE_OUT**, **RESIZE_LESS**.

Default: true

producing_operation -- *a Operation field of model*

PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK

The **Operation** to be created in order to produce additional flow into this Buffer as needed due to consuming **Operations**.

It is a Field_Error for this field to be [unspecified], the default, if any **Buffer_Plan** needs to issue a producing **Operation_Plan**.

Default: [unspecified]

flow_plan_selection_start -- *a Expression field of model*

PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK

This expression is called before **Flow_Plan**s are selected in order to build bags. For the convenience of OIL, expression writer following special variables are available,

buffer_plan: : Bound to Buffer_Plan of this Buffer, problem_period : Bound to the problem period of the currently selected problem, problem_category : Bound to Problem_Category of the currently selected problem active_strategy : Bound to Active_Strategy solving problem on this Buffer.

Example Usage:
Default: [unspecified]

flow_plan_selection_end -- a Expression field of model
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK

This expression is called after Flow_Plans are selected to build the bags. For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to Buffer_Plan of this Buffer, problem_period : Bound to the problem period of the currently selected problem, problem_category : Bound to Problem_Category of the currently selected problem active_strategy : Bound to Active_Strategy solving problem on this Buffer.

Example Usage:
Default: [unspecified]

flow_plan_selection_interval -- a Expression field of model
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK

This expression returns the Date_Range over which resolver will look to select Flow_Plans in order to build bags. For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to Buffer_Plan of this Buffer, problem_period : Bound to the problem period of the currently selected problem, problem_category : Bound to Problem_Category of the currently selected problem change_category : Bound to Strategy_Change.category for which resolver is building bag, active_strategy : Bound to Active_Strategy solving problem on this Buffer.

Example Usage:
buffer_b1 = supply_chains.find("Supl").sites.find("Site").buffers.find(Buffer_b1.set_flow_plan_selection_interval("Date_Range(problem_start - 1 Week, problem_end)")
Default: problem_period

flow_plan_filter -- a Expression field of model
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK

This expression takes following parameters and if it returns true, the Flow_Plan will be selected and put into the bag. If it returns false, the Flow_Plan will not be put into the bag.

For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to Buffer_Plan of this Buffer, problem_period : Bound to the problem period of the currently selected problem, problem_category : Bound to Problem_Category of the currently selected problem change_category : Bound to Strategy_Change.category for which resolver is building bag, active_strategy : Bound to Active_Strategy solving problem on this Buffer, flow_plan : Bound to Flow_Plan while doing bag candidate selection.

Example Usage:
Default: true

flow_plan_rank -- a Expression field of model
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK

This expression returns the rank of the Flow_Plan which is used by the resolver in selecting the Flow_Plan. Higher 'rank' means that the Flow_Plan is more likely to be chosen later from the bag.

For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to Buffer_Plan of this Buffer, problem_period : Bound to the problem period of the currently selected problem, problem_category : Bound to Problem_Category of the currently selected problem change_category : Bound to Strategy_Change.category for which resolver is building bag, active_strategy : Bound to Active_Strategy solving problem on this Buffer, flow_plan : Bound to Flow_Plan while doing bag candidate selection.

Example Usage:
Default: 1.0

Extensions *PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK* Extension

continue *Flow_Plan_selection* -- *a Expression field of model*
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK
 This expression takes following parameters and if it returns true, the selection of *Flow_Plan* will continue to be put into the bag. If it returns false, further *Flow_Plan*s will not be selected.

For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to *Buffer_Plan* of this *Buffer*; *problem_period* : Bound to the *problem_period* of the currently selected *problem*; *problem_category* : Bound to *Problem_Category* of the currently selected *problem*; *change_category* : Bound to *Strategy_Change_category* for which resolver is building bag; *active_strategy* : Bound to *Active_Strategy* solving problem on this *Buffer*; *flow_plan* : Bound to *Flow_Plan* while doing bag candidate selection.

Example Usage:
 Default: true

flow_plan_resize_quantity_range -- *a Expression field of model*
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK
 Replan restrictions This expression returns the *Quantity_Range* within which the selected *Flow_Plan* can be resized by the resolver.

For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to *Buffer_Plan* of this *Buffer*; *problem_period* : Bound to the *problem_period* of the currently selected *problem*; *problem_category* : Bound to *Problem_Category* of the currently selected *problem*; *change_category* : Bound to *Strategy_Change_category* for which resolver is building bag; *active_strategy* : Bound to *Active_Strategy* solving problem on this *Buffer*; *flow_plan* : Bound to *Flow_Plan* while doing bag candidate selection.

Example Usage:
 Default: [unspecified]

flow_plan_move_restriction -- *a Expression field of model*
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK
 This expression returns the *Date_Range* within which the selected *Flow_Plan* can be moved by the resolver.

Extensions *SUPPLY_CALENDAR* Extension

For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to *Buffer_Plan* of this *Buffer*; *problem_period* : Bound to the *problem_period* of the currently selected *problem*; *problem_category* : Bound to *Problem_Category* of the currently selected *problem*; *change_category* : Bound to *Strategy_Change_category* for which resolver is building bag; *active_strategy* : Bound to *Active_Strategy* solving problem on this *Buffer*; *flow_plan* : Bound to *Flow_Plan* while doing bag candidate selection.

Example Usage:
 Default: [unspecified]

flow_plan_split_restriction -- *a Expression field of model*
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK
 This expression returns the *Quantity_Range* within which the selected *Flow_Plan* can be split by the resolver for a move

For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to *Buffer_Plan* of this *Buffer*; *problem_period* : Bound to the *problem_period* of the currently selected *problem*; *problem_category* : Bound to *Problem_Category* of the currently selected *problem*; *change_category* : Bound to *Strategy_Change_category* for which resolver is building bag; *active_strategy* : Bound to *Active_Strategy* solving problem on this *Buffer*; *flow_plan* : Bound to *Flow_Plan* while doing bag candidate selection.

Example Usage:
 Default: [unspecified]

SUPPLY_CALENDAR -- *a flow_policy extension of model* Buffer

A **SUPPLY_CALENDAR** Buffer is replenished artificially as specified in a *Calendar*. The *calendar* specifies additional supply "flows" producing into the Buffer. This can be used as a simple way to model regular deliveries from a reliable supplier. No Operations are generated; no capacity or items are consumed; and no purchase Requests. The buffer is just replenished "spontaneously", on its own.

It may also be used to model other requirements that are easily modeled as an additional material requirement. For example, the number of starts into a facility may be limited to a certain number each month. The easiest way to model that may be as an additional item that is needed by the first Operation. The quantity of that item available each month limits the number of starts. Note that unused starts from the previous months will carry over. See the ON_HAND_CALENDAR_flow_policy for the same that does not carry over.

The SUPPLY_CALENDAR model has fields that reference these models :

Calendar:

calendar -- a Calendar field of model SUPPLY_CALENDAR

A Calendar of QUANTITY entries that each specify a supply producing into the Buffer. Unconsumed supply accumulates in the Buffer. Note that any default_quantity set for this calendar will be ignored, since its impact on on_hand is impossible to determine without a day_pattern.

Default: [unspecified]

ON_HAND_CALENDAR -- a flow_policy extension of model Buffer

A ON_HAND_CALENDAR Buffer is replenished artificially as specified in a Calendar. The calendar specifies the new on_hand balance at that Date (as if whatever flow is necessary to give that balance will arrive or depart at that Date). This can be used as a simple way to model regular deliveries from a reliable supplier. No Operations are generated; no capacity or items are consumed; and no purchase Requests. The buffer is just replenished "spontaneously", on its own.

It may also be used to model other requirements that are easily modeled as an additional material requirement. For example, the number of starts into a facility may be limited to a certain number each month. The easiest way to model that may be as an additional item that is needed by the first Operation. The quantity of that item available each month limits the number of starts. Note that unused starts from the previous months will not carry over. See the SUPPLY_CALENDAR_flow_policy for the same that does carry over.

The ON_HAND_CALENDAR model has fields that reference these models :

Calendar:

calendar -- a Calendar field of model ON_HAND_CALENDAR

A Calendar of QUANTITY entries that each specify a new on-hand balance for the Buffer. Production or consumption occurs spontaneously such that the specified balance occurs. Note that any default_quantity set for this calendar will be ignored, since its impact on on_hand is impossible to determine without a day_pattern.

Default: [unspecified]

ON_HAND_CALENDAR_FILTER_AND_RANK -- a flow_policy extension of model Buffer

A ON_HAND_CALENDAR Buffer is replenished artificially as specified in a Calendar. The calendar specifies the new on_hand balance at that Date (as if whatever flow is necessary to give that balance will arrive or depart at that Date). This can be used as a simple way to model regular deliveries from a reliable supplier. No Operations are generated; no capacity or items are consumed; and no purchase Requests. The buffer is just replenished "spontaneously", on its own.

It may also be used to model other requirements that are easily modeled as an additional material requirement. For example, the number of starts into a facility may be limited to a certain number each month. The easiest way to model that may be as an additional item that is needed by the first Operation. The quantity of that item available each month limits the number of starts. Note that unused starts from the previous months will not carry over. See the SUPPLY_CALENDAR_flow_policy for the same that does carry over.

The ON_HAND_CALENDAR_FILTER_AND_RANK model has fields that reference these models :

Calendar:

calendar -- a Calendar field of model

ON_HAND_CALENDAR_FILTER_AND_RANK

A Calendar of QUANTITY entries that each specify a new on-hand balance for the Buffer. Production or consumption occurs spontaneously such that the specified balance occurs. Note that any default_quantity set for this calendar will be ignored, since its impact on on_hand is impossible to determine without a day_pattern.

Default: [unspecified]

flow_plan_selection_start -- a Expression field of model

ON_HAND_CALENDAR_FILTER_AND_RANK

This expression is called before Flow_Plan is selected in order to build bags. For the convenience of OIL expression writer following special variables are available.

Extensions	ON_HAND_CALENDAR_FILTER_AND_RANK Extension
------------	--

buffer_plan: : Bound to Buffer_Plan of this Buffer; **problem_period**: : Bound to the problem period of the currently selected problem; **problem_category**: : Bound to Problem_Category of the currently selected problem; **active_strategy**: : Bound to Active_Strategy solving problem on this Buffer.

Example Usage:
Default: [unspecified]

flow_plan_selection_end - - *a Expression field of model*
ON_HAND_CALENDAR_FILTER_AND_RANK
This expression is called after Flow_Plans are selected to build the bags. For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to Buffer_Plan of this Buffer; **problem_period**: : Bound to the problem period of the currently selected problem; **problem_category**: : Bound to Problem_Category of the currently selected problem; **active_strategy**: : Bound to Active_Strategy solving problem on this Buffer.

Example Usage:
Default: [unspecified]

flow_plan_selection_interval - - *a Expression field of model*
ON_HAND_CALENDAR_FILTER_AND_RANK
This expression returns the Date_Range over which resolver will look to select Flow_Plans in order to build bags. For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to Buffer_Plan of this Buffer; **problem_period**: : Bound to the problem period of the currently selected problem; **problem_category**: : Bound to Problem_Category of the currently selected problem; **change_category**: : Bound to Strategy_Change.category for which resolver is building bag; **active_strategy**: : Bound to Active_Strategy solving problem on this Buffer.

Example Usage:

```
Buffer b1 = supply_chains.find("Sup1").sites.find("Site1").buffers.find(Buffer1)
b1.set_flow_plan_selection_interval("Date_Range(problem_start - 1 Week, problem_end)")
Default: problem_period
```

Extensions	ON_HAND_CALENDAR_FILTER_AND_RANK Extension
------------	--

flow_plan_filter - - *a Expression field of model*
ON_HAND_CALENDAR_FILTER_AND_RANK
This expression takes following parameters and if it returns true, the Flow_Plan will be selected and put into the bag. If it returns false, the Flow_Plan will not be put into the bag.

For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to Buffer_Plan of this Buffer; **problem_period**: : Bound to the problem period of the currently selected problem; **problem_category**: : Bound to Problem_Category of the currently selected problem; **change_category**: : Bound to Strategy_Change.category for which resolver is building bag; **active_strategy**: : Bound to Active_Strategy solving problem on this Buffer; **flow_plan**: : Bound to Flow_Plan while doing bag candidate selection.

Example Usage:
Default: true

flow_plan_rank - - *a Expression field of model*
ON_HAND_CALENDAR_FILTER_AND_RANK
This expression returns the rank of the Flow_Plan which is used by the resolver in selecting the Flow_Plan. Higher 'rank' means that the Flow_Plan is more likely to be chosen later from the bag.

For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to Buffer_Plan of this Buffer; **problem_period**: : Bound to the problem period of the currently selected problem; **problem_category**: : Bound to Problem_Category of the currently selected problem; **change_category**: : Bound to Strategy_Change.category for which resolver is building bag; **active_strategy**: : Bound to Active_Strategy solving problem on this Buffer; **flow_plan**: : Bound to Flow_Plan while doing bag candidate selection.

Example Usage:
Default: 1.0

Extensions	ON_HAND_CALENDAR_FILTER_AND_RANK Extension
------------	--

continue *flow_plan.selection* -- *a Expression field of model*
ON_HAND_CALENDAR_FILTER_AND_RANK
This expression takes following parameters and if it returns true, the selection of *Flow_Plan* will continue to be put into the bag. If it returns false, further *Flow_Plan*s will not be selected.

For the convenience of OIL expression writer following special variables are available.

buffer_plan: : Bound to *Buffer_Plan* of this *Buffer.problem.period* : Bound to the problem period of the currently selected problem. *problem_category* : Bound to *Problem.Category* of the currently selected problem *change_category* : Bound to *Strategy.Change.category* for which resolver is building bag. *active_strategy* : Bound to *Active_Strategy* solving problem on this *Buffer.flow_plan* : Bound to *Flow_Plan* while doing bag candidate selection.

Example Usage:
Default: true

flow_plan.resize.quantity_range -- *a Expression field of model*
ON_HAND_CALENDAR_FILTER_AND_RANK
Replan restrictions This expression returns the *Quantity.Range* within which the selected *Flow_Plan* can be resized by the resolver.

For the convenience of OIL expression writer following special variables are available.

buffer_plan: : Bound to *Buffer_Plan* of this *Buffer.problem.period* : Bound to the problem period of the currently selected problem. *problem_category* : Bound to *Problem.Category* of the currently selected problem *change_category* : Bound to *Strategy.Change.category* for which resolver is building bag. *active_strategy* : Bound to *Active_Strategy* solving problem on this *Buffer.flow_plan* : Bound to *Flow_Plan* while doing bag candidate selection.

Example Usage:
Default: [unspecified]

flow_plan.move.restriction -- *a Expression field of model*
ON_HAND_CALENDAR_FILTER_AND_RANK
This expression returns the *Date.Range* within which the selected *Flow_Plan* can be moved by the resolver. For the convenience of OIL expression writer following special variables are available.

Extensions	FLOW_LIMIT_CALENDAR Extension
------------	-------------------------------

buffer_plan: : Bound to *Buffer_Plan* of this *Buffer.problem.period* : Bound to the problem period of the currently selected problem. *problem_category* : Bound to *Problem.Category* of the currently selected problem *change_category* : Bound to *Strategy.Change.category* for which resolver is building bag. *active_strategy* : Bound to *Active_Strategy* solving problem on this *Buffer.flow_plan* : Bound to *Flow_Plan* while doing bag candidate selection.

Example Usage:
Default: [unspecified]

flow_plan.split.restriction -- *a Expression field of model*
ON_HAND_CALENDAR_FILTER_AND_RANK
This expression returns the *Quantity.Range* within which the selected *Flow_Plan* can be split by the resolver for a move

For the convenience of OIL expression writer following special variables are available.

buffer_plan: : Bound to *Buffer_Plan* of this *Buffer.problem.period* : Bound to the problem period of the currently selected problem. *problem_category* : Bound to *Problem.Category* of the currently selected problem *change_category* : Bound to *Strategy.Change.category* for which resolver is building bag. *active_strategy* : Bound to *Active_Strategy* solving problem on this *Buffer.flow_plan* : Bound to *Flow_Plan* while doing bag candidate selection.

Example Usage:
Default: [unspecified]

FLOW_LIMIT_CALENDAR -- a flow_policy extension of model Buffer

A *FLOW_LIMIT_CALENDAR* Buffer has a behaviour identical to a *ON_HAND_CALENDAR* except that the *NEGATIVE_ON_HAND* problems are in this case called *OVER_FLOW_LIMIT* problems. This can be used to model unit capacity buffers. See *ON_HAND_CALENDAR* for more details on how this flow policy is supposed to work.

The *FLOW_LIMIT_CALENDAR* model has fields that references these models : *Calendar*.

Extensions	FLOW_LIMIT_CALENDAR_FILTER_AND_RANK Extension
------------	---

calendar - - - *a Calendar field of model* **FLOW_LIMIT_CALENDAR_FILTER_AND_RANK**

A Calendar of QUANTITY entries that each specify a new on-hand balance for the Buffer. Supply or consumption occurs spontaneously such that the specified balance occurs.

Default: [unspecified]

FLOW_LIMIT_CALENDAR_FILTER_AND_RANK - - *a flow_policy extension of model* **Buffer**

A FLOW_LIMIT_CALENDAR Buffer has a behaviour identical to a ON_HAND_CALENDAR except that the NEGATIVE_ON_HAND problems are in this case called OVER_FLOW_LIMIT problems. This can be used to model unit capacity buffers. See ON_HAND_CALENDAR for more details on how this flow policy is supposed to work.

The FLOW_LIMIT_CALENDAR_FILTER_AND_RANK model has fields that references these models :

Calendar:

calendar - - - *a Calendar field of model*

FLOW_LIMIT_CALENDAR_FILTER_AND_RANK

A Calendar of QUANTITY entries that each specify a new on-hand balance for the Buffer. Supply or consumption occurs spontaneously such that the specified balance occurs.

Default: [unspecified]

flow_plan_selection_start - - - *a Expression field of model*

FLOW_LIMIT_CALENDAR_FILTER_AND_RANK

This expression is called before Flow_Plan is selected in order to build bags. For the convenience of OIL expression writer following special variables are available,

buffer_plan : : Bound to Buffer_Plan of this Buffer. **problem_period** : Bound to the problem period of the currently selected problem. **problem_category** : Bound to Problem_Category of the currently selected problem. **active_strategy** : Bound to Active_Strategy solving problem on this Buffer.

Example Usage:
Default: [unspecified]

Extensions	FLOW_LIMIT_CALENDAR_FILTER_AND_RANK Extension
------------	---

flow_plan_selection_end - - - *a Expression field of model*

FLOW_LIMIT_CALENDAR_FILTER_AND_RANK

This expression is called after Flow_Plan is selected to build the bags. For the convenience of OIL expression writer following special variables are available,

buffer_plan : : Bound to Buffer_Plan of this Buffer. **problem_period** : Bound to the problem period of the currently selected problem. **problem_category** : Bound to Problem_Category of the currently selected problem. **active_strategy** : Bound to Active_Strategy solving problem on this Buffer.

Example Usage:
Default: [unspecified]

flow_plan_selection_interval - - - *a Expression field of model*

FLOW_LIMIT_CALENDAR_FILTER_AND_RANK

This expression returns the Date_Range over which resolver will look to select Flow_Plan in order to build bags. For the convenience of OIL expression writer following special variables are available.

buffer_plan : : Bound to Buffer_Plan of this Buffer. **problem_period** : Bound to the problem period of the currently selected problem. **problem_category** : Bound to Problem_Category of the currently selected problem. **change_category** : Bound to Strategy_Change.category for which resolver is building bag. **active_strategy** : Bound to Active_Strategy solving problem on this Buffer.

Example Usage:

Buffer.b1 = supply_chains.find("Sup1").sites.find("Site1").buffers.find(Buffer1).set(flow_plan_selection_interval("Date_Range(problem_start - 1 Week, problem_end)"))

Default: problem_period

flow_plan_filter - - - *a Expression field of model*

FLOW_LIMIT_CALENDAR_FILTER_AND_RANK

This expression takes following parameters and if it returns true, the Flow_Plan will be selected and put into the bag. If it returns false, the Flow_Plan will not be put into the bag.

For the convenience of OIL expression writer following special variables are available.

Extensions	FLOW_LIMIT_CALENDAR_FILTER_AND_RANK Extension
------------	---

buffer_plan: : Bound to Buffer_Plan of this Buffer.problem_period : Bound to the problem period of the currently selected problem.problem_category : Bound to Problem_Category of the currently selected problem.change_category : Bound to Strategy_Change.category for which resolver is building bag.active_strategy : Bound to Active_Strategy solving problem on this Buffer.flow_plan : Bound to Flow_Plan while doing bag candidate selection.

Example Usage:
Default: true

flow_plan_rank : - a Expression field of model
FLOW_LIMIT_CALENDAR_FILTER_AND_RANK

This expression returns the rank of the Flow_Plan which is used by the resolver in selecting the Flow_Plan. Higher rank means that the Flow_Plan is more likely to be chosen later from the bag.

For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to Buffer_Plan of this Buffer.problem_period : Bound to the problem period of the currently selected problem.problem_category : Bound to Problem_Category of the currently selected problem.change_category : Bound to Strategy_Change.category for which resolver is building bag.active_strategy : Bound to Active_Strategy solving problem on this Buffer.flow_plan : Bound to Flow_Plan while doing bag candidate selection.

Example Usage:
Default: 1.0

continue_flow_plan_selection : - a Expression field of model
FLOW_LIMIT_CALENDAR_FILTER_AND_RANK

This expression takes following parameters and if it returns true, the selection of Flow_Plan will continue to be put into the bag. If it returns false, further Flow_Plans will not be selected.

For the convenience of OIL expression writer following special variables are available.

Extensions	FLOW_LIMIT_CALENDAR_FILTER_AND_RANK Extension
------------	---

buffer_plan: : Bound to Buffer_Plan of this Buffer.problem_period : Bound to the problem period of the currently selected problem.problem_category : Bound to Problem_Category of the currently selected problem.change_category : Bound to Strategy_Change.category for which resolver is building bag.active_strategy : Bound to Active_Strategy solving problem on this Buffer.flow_plan : Bound to Flow_Plan while doing bag candidate selection.

Example Usage:
Default: true

flow_plan_resize_quantity_range : - a Expression field of model
FLOW_LIMIT_CALENDAR_FILTER_AND_RANK

Replan restrictions This expression returns the Quantity_Range within which the selected Flow_Plan can be resized by the resolver.

For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to Buffer_Plan of this Buffer.problem_period : Bound to the problem period of the currently selected problem.problem_category : Bound to Problem_Category of the currently selected problem.change_category : Bound to Strategy_Change.category for which resolver is building bag.active_strategy : Bound to Active_Strategy solving problem on this Buffer.flow_plan : Bound to Flow_Plan while doing bag candidate selection.

Example Usage:
Default: (unspecified)

flow_plan_move_restriction : - a Expression field of model
FLOW_LIMIT_CALENDAR_FILTER_AND_RANK

This expression returns the Date_Range within which the selected Flow_Plan can be moved by the resolver. For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to Buffer_Plan of this Buffer.problem_period : Bound to the problem period of the currently selected problem.problem_category : Bound to Problem_Category of the currently selected problem.change_category : Bound to Strategy_Change.category for which resolver is building bag.active_strategy : Bound to Active_Strategy solving problem on this Buffer.flow_plan : Bound to Flow_Plan while doing bag candidate selection.

Example Usage:

Extensions	INFINITE Extension
------------	--------------------

Default: [unspecified]

Flow_plan_split_restriction -- *a Expression field of model FLOW_LIMIT_CALENDAR_FILTER_AND_RANK*

This expression returns the Quantity_Range within which the selected Flow_Plan can be split by the resolver for a move

For the convenience of OIL expression writer following special variables are available,

buffer_plan : Bound to Buffer_Plan of this Buffer. **problem_period** : Bound to the problem period of the currently selected problem. **problem_category** : Bound to Problem_Category of the currently selected problem. **change_category** : Bound to Strategy_Change_category for which resolver is building bag. **active_strategy** : Bound to Active_Strategy solving problem on this Buffer. **flow_plan** : Bound to Flow_Plan while doing bag candidate selection.

Example Usage:
Default: [unspecified]

INFINITE -- a flow_policy extension of model Buffer

An INFINITE Buffer models the flows that are produced into or consumed from it, but detects no Problems and will never create Operations to replenish itself. It allows you to display the flow of material, without imposing any constraints on the plan due to that flow.

This is often used for raw materials where the purchasing of those materials is not planned by this system. For example, an inventory that is effectively managed via reorder-point can be modelled in this system as an INFINITE Buffer.

Note that this can model not only the infinite supply of an abundant material, but also the infinite disposal or consumption of an uninteresting material.

Problems Identified and Resolved by INFINITE buffer:

INFINITE flow policy does not identify any problems. The only time an INFINITE buffer will identify problems is when user has attached Buffer_Problem. Detectors to the buffer. These problem detectors will identify problems regardless of flow policy.

SUPPLIER -- *a flow_policy extension of model Buffer*

Extensions	BASIC Extension
------------	-----------------

A SUPPLIER Buffer models flows into the system from an external (not modelled) source. Requests which originate from downstream are satisfied if the material requirement date is at or after the supplier fence, and not satisfied if the request is nearer the current date. In that sense, this is like an INFINITE flow policy, with the additional restriction that it is only INFINITE at and after the fence.

This is used to model orders from suppliers with restrictions on when an order can be processed.

This is often used for raw materials where the purchasing of those materials is not planned by this system. For example, an inventory that is effectively managed via reorder-point but requires advance purchase planning can be modelled in this system as a SUPPLIER Buffer.

fence -- *a Horizon_Date field of model SUPPLIER*
At and after the fence, this flow policy behaves exactly like an INFINITE flow policy. Currently, inside the fence, no requests are satisfied.
Default: 00

BASIC -- a flow_policy extension of model Buffer

A BASIC Buffer maintains a min_time safety time and a min_on_hand safety quantity. The min_on_hand gives a fixed minimum level for the safety stock. The min_time gives a fixed time of consumption that should always be covered (for example, setting min_time to "3 weeks" will keep enough stock to cover the next 3 weeks of out flows).

Up to excess_on_hand more than the minimum required is allowed in the Buffer. By setting excess_on_hand larger than zero, planning instability is reduced, and planning speed is increased. However, keeping it smaller minimizes excess inventory.

For example, consider a Buffer with min_on_hand set to "100", min_time set to "2 weeks", and excess_on_hand set to "20". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the on_hand maintained until 4/1 will be "100", the min_on_hand. From 4/1 to 4/15, "200" will be maintained in order to cover the next "2 weeks" of out flows. If the out flow is reduced to "190", the producing operation plans may not change -- since "200" is only "10" greater than "190", and "10" is less than "20", the excess_on_hand that is allowed.

The BASIC model has fields that references these models:
Operation.

Extensions**BASIC_FILTER_AND_RANK Extension****min_time** -- *a Time field of model BASIC*

Items are planned such that they arrive at least 'min_time' earlier than otherwise needed. This may be used as a safety time -- the next 'min_time' of consumption will be maintained in the Buffer. 'min_time' is represented internally as seconds so if specified as days, weeks, years, its will be converted to seconds. This needs to be considered around daylight savings time boundaries.

Default: 0

min_on_hand -- *a Quantity field of model BASIC*

The minimum on_hand balance that should be present in this Buffer. This Quantity is converted to the unit of this Buffer.

Default: 0

excess_on_hand -- *a Quantity field of model BASIC*

Up to this Quantity more than the minimum required is allowed in this Buffer. Setting this above zero reduces instability, and speeds planning. Setting it smaller decreases excess inventory levels. This Quantity is converted to the unit of this Buffer.

Default: 0

producing_operation -- *a Operation field of model BASIC*

The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field_Error for this field to be [unspecified], the default, if any Buffer_Plan needs to issue a producing Operation_Plan.

Default: [unspecified]

supplying_operation -- *a Operation field of model BASIC*

Obsolete! This field has been renamed 'producing_operation' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.

Default: [unspecified]

Properties: obsolete=True

BASIC_FILTER_AND_RANK -- *a Flow_policy extension of model Buffer*

Just like Basic Flow Policy, but with user entry points to control resolver behavior

The BASIC_FILTER_AND_RANK model has fields that references these models : Operation.

Extensions**BASIC_FILTER_AND_RANK Extension****min_time** -- *a Time field of model BASIC_FILTER_AND_RANK*

Items are planned such that they arrive at least 'min_time' earlier than otherwise needed. This may be used as a safety time -- the next 'min_time' of consumption will be maintained in the Buffer. 'min_time' is represented internally as seconds so if specified as days, weeks, years, its will be converted to seconds. This needs to be considered around daylight savings time boundaries.

Default: 0

min_on_hand -- *a Quantity field of model BASIC_FILTER_AND_RANK*

The minimum on_hand balance that should be present in this Buffer. This Quantity is converted to the unit of this Buffer.

Default: 0

excess_on_hand -- *a Quantity field of model BASIC_FILTER_AND_RANK*

Up to this Quantity more than the minimum required is allowed in this Buffer. Setting this above zero reduces instability, and speeds planning. Setting it smaller decreases excess inventory levels. This Quantity is converted to the unit of this Buffer.

Default: 0

producing_operation -- *a Operation field of model***BASIC_FILTER_AND_RANK**

The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field_Error for this field to be [unspecified], the default, if any Buffer_Plan needs to issue a producing Operation_Plan.

Default: [unspecified]

supplying_operation -- *a Operation field of model***BASIC_FILTER_AND_RANK**

Obsolete! This field has been renamed 'producing_operation' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.

Default: [unspecified]

Properties: obsolete=True

flow_plan_selection_start -- *a Expression field of model***BASIC_FILTER_AND_RANK**

This expression is called before Flow_Plans are selected in order to build bags. For the convenience of OIL expression writer following special variables are available.

buffer_plan : Bound to Buffer_Plan of this Buffer. **problem_period** : Bound to the problem period of the currently selected problem. **problem_category** : Bound to Problem_Category of the currently selected problem. **active_strategy** : Bound to Active_Strategy solving problem on this Buffer.

Example Usage:

Default: [unspecified]

flow_plan_selection_end - - *a Expression field of model*

BASIC_FILTER_AND_RANK

This expression is called after Flow_Plans are selected to build the bags. For the convenience of OIL expression writer following special variables are available,

buffer_plan : Bound to Buffer_Plan of this Buffer. **problem_period** : Bound to the problem period of the currently selected problem. **problem_category** : Bound to Problem_Category of the currently selected problem. **active_strategy** : Bound to Active_Strategy solving problem on this Buffer.

Example Usage:

Default: [unspecified]

flow_plan_selection_interval - - *a Expression field of model*

BASIC_FILTER_AND_RANK

This expression returns the Date_Range over which resolver will look to select Flow_Plans in order to build bags. For the convenience of OIL expression writer following special variables are available,

buffer_plan : Bound to Buffer_Plan of this Buffer. **problem_period** : Bound to the problem period of the currently selected problem. **problem_category** : Bound to Problem_Category of the currently selected problem. **change_category** : Bound to Strategy_Change.category for which resolver is building bag. **active_strategy** : Bound to Active_Strategy solving problem on this Buffer.

Example Usage:

```
Buffer b1 = supply_chains.find("Sup1").sites.find("Site1").buffers.find(Buffer)
b1.set_flow_plan_selection_interval("Date_Range(problem_start - 1 Week,
problem_end)")
Default: problem_period
```

flow_plan_filter - - *a Expression field of model* **BASIC_FILTER_AND_RANK**
This expression takes following parameters and if it returns true, the Flow_Plan will be selected and put into the bag. If it returns false, the Flow_Plan will not be put into the bag.

For the convenience of OIL expression writer following special variables are available,

buffer_plan : Bound to Buffer_Plan of this Buffer. **problem_period** : Bound to the problem period of the currently selected problem. **problem_category** : Bound to Problem_Category of the currently selected problem. **change_category** : Bound to Strategy_Change.category for which resolver is building bag. **active_strategy** : Bound to Active_Strategy solving problem on this Buffer. **flow_plan** : Bound to Flow_Plan while doing bag candidate selection.

Example Usage:

Default: true

flow_plan_rank - - *a Expression field of model* **BASIC_FILTER_AND_RANK**
This expression returns the rank of the Flow_Plan which is used by the resolver in selecting the Flow_Plan. Higher rank means that the Flow_Plan is more likely to be chosen later from the bag.

For the convenience of OIL expression writer following special variables are available,

buffer_plan : Bound to Buffer_Plan of this Buffer. **problem_period** : Bound to the problem period of the currently selected problem. **problem_category** : Bound to Problem_Category of the currently selected problem. **change_category** : Bound to Strategy_Change.category for which resolver is building bag. **active_strategy** : Bound to Active_Strategy solving problem on this Buffer. **flow_plan** : Bound to Flow_Plan while doing bag candidate selection.

Example Usage:

Default: 1.0

continue_flow_plan_selection - - *a Expression field of model*
BASIC_FILTER_AND_RANK

This expression takes following parameters and if it returns true, the selection of Flow_Plan will continue to be put into the bag. If it returns false, further Flow_Plans will not be selected.

For the convenience of OIL expression writer following special variables are available.

buffer_plan : Bound to Buffer_Plan of this Buffer. **problem_period :** Bound to the problem period of the currently selected problem. **problem_category :** Bound to Problem_Category of the currently selected problem. **change_category :** Bound to Strategy_Change.category for which resolver is building bag. **active_strategy :** Bound to Active_Strategy solving problem on this Buffer. **flow_plan :** Bound to Flow_Plan while doing bag candidate selection.

Example Usage:
Default: [unspecified]

flow_plan_resize_quantity_range -- *a Expression field of model*
BASIC_FILTER_AND_RANK

Replan restrictions This expression returns the Quantity_Range within which the selected Flow_Plan can be resized by the resolver.

For the convenience of OIL expression writer following special variables are available.

buffer_plan : Bound to Buffer_Plan of this Buffer. **problem_period :** Bound to the problem period of the currently selected problem. **problem_category :** Bound to Problem_Category of the currently selected problem. **change_category :** Bound to Strategy_Change.category for which resolver is building bag. **active_strategy :** Bound to Active_Strategy solving problem on this Buffer. **flow_plan :** Bound to Flow_Plan while doing bag candidate selection.

Example Usage:
Default: [unspecified]

flow_plan_move_restriction -- *a Expression field of model*
BASIC_FILTER_AND_RANK

This expression returns the Date_Range within which the selected Flow_Plan can be moved by the resolver.

For the convenience of OIL expression writer following special variables are available.

buffer_plan : Bound to Buffer_Plan of this Buffer. **problem_period :** Bound to the problem period of the currently selected problem. **problem_category :** Bound to Problem_Category of the currently selected problem. **change_category :** Bound to Strategy_Change.category for which resolver is building bag. **active_strategy :** Bound to Active_Strategy solving problem on this Buffer. **flow_plan :** Bound to Flow_Plan while doing bag candidate selection.

Example Usage:
Default: [unspecified]

flow_plan_split_restriction -- *a Expression field of model*
BASIC_FILTER_AND_RANK

This expression returns the Quantity_Range within which the selected Flow_Plan can be split by the resolver for a move

For the convenience of OIL expression writer following special variables are available.

buffer_plan : Bound to Buffer_Plan of this Buffer. **problem_period :** Bound to the problem period of the currently selected problem. **problem_category :** Bound to Problem_Category of the currently selected problem. **change_category :** Bound to Strategy_Change.category for which resolver is building bag. **active_strategy :** Bound to Active_Strategy solving problem on this Buffer. **flow_plan :** Bound to Flow_Plan while doing bag candidate selection.

Example Usage:
Default: [unspecified]

FIXED_QUANTITY -- a flow_policy extension of model Buffer

A **FIXED_QUANTITY** Buffer issues replenishment Operations all with the same 'quantity'. The Operations are planned to arrive such that the Buffer's on_hand balance never drops below 'min_on_hand'. This policy is often called 'Fixed Order Quantity (FOQ)'.

Up to 'excess_on_hand' more than the minimum required is allowed in the Buffer. By setting 'excess_on_hand' larger than zero, planning instability is reduced, and planning speed is increased. However, keeping it smaller minimizes excess inventory.

The **FIXED_QUANTITY** model has fields that references these models :
Operation.

min_on_hand - - *a Quantity field of model FIXED_QUANTITY*

The minimum on_hand balance that should be present in this Buffer. This Quantity is converted to the unit of this Buffer.

Default: 0

min_time - - *a Time field of model FIXED_QUANTITY*

Items are planned such that they arrive at least min_time earlier than otherwise needed. This may be used as a safety time -- the next min_time of consumption will be maintained in the Buffer. min_time is represented internally as seconds so if specified as days, weeks, years, its will be converted to seconds. This needs to be considered around daylight savings time boundaries.

Default: 0

excess_on_hand - - *a Quantity field of model FIXED_QUANTITY*

Up to this Quantity more than the minimum required is allowed in this Buffer. Setting this above zero reduces instability, and speeds planning. Setting it smaller decreases excess inventory levels. This Quantity is converted to the unit of this Buffer.

Default: 0

quantity - - *a Quantity field of model FIXED_QUANTITY*

The fixed Quantity of the Item to be produced by each producing_operation that is issued by this FIXED_QUANTITY Buffer. This Quantity is converted to the preferred_measure of this Buffer. The default is "1", which means 1 unit of the Buffer.

Default: 1

producing_operation - - *a Operation field of model FIXED_QUANTITY*

The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field_Error for this field to be [unspecified], the default, if any Buffer_Plan needs to issue a producing Operation_Plan.

Default: [unspecified]

supplying_operation - - *a Operation field of model FIXED_QUANTITY*

Obsolete! This field has been renamed producing_operation in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.

Default: [unspecified]

Properties: obsolete=True

MULTIPLE - - *a flow_policy extension of model Buffer*

A MULTIPLE Buffer issues replenishment Operations whose sizes are integral multiples of multiple_quantity and are bounded by quantity_range. The Operations are planned to arrive such that the Buffer's on_hand balance never drops below min_on_hand.

Up to 'excess_on_hand' more than the minimum required is allowed in the Buffer. By setting 'excess_on_hand' larger than zero, planning instability is reduced, and planning speed is increased. However, keeping it smaller minimizes excess inventory.

The MULTIPLE model has fields that references these models :
Operation.

min_on_hand - - *a Quantity field of model MULTIPLE*

The minimum on_hand balance that should be present in this Buffer. This Quantity is converted to the unit of this Buffer.

Default: 0

excess_on_hand - - *a Quantity field of model MULTIPLE*

Up to this Quantity more than the minimum required is allowed in this Buffer. Setting this above zero reduces instability, and speeds planning. Setting it smaller decreases excess inventory levels. This Quantity is converted to the unit of this Buffer.

Default: 0

quantity_range - - *a Quantity_Range field of model MULTIPLE*

The range between which supplying operations of the buffer are constrained to lie.

Note that the smallest legal size of a supplying operation can be more than the min of the quantity_range, since it has to be a multiple of multiple_quantity. Similarly, the largest legal size can be smaller than the max of the quantity_range.

Default: [1, 00]

multiple_quantity - - *a Quantity field of model MULTIPLE*

The Quantity of which the Quantity of the Item to be produced by each producing_operation that is issued by this MULTIPLE Buffer is a multiple of. This Quantity is converted to the unit of this Buffer. If multiple_quantity is set to "0", it implies that the quantities of producing operations can be any number (integer or non-integer) in the range of "quantity_range". The default is "0".

Default: 0

min_time -- *a Time field of model MULTIPLE*

Items are planned such that they arrive at least min_time earlier than otherwise needed. This may be used as a safety time -- the next min_time of consumption will be maintained in the Buffer. min_time is represented internally as seconds so it specified as days, weeks, years, its will be converted to seconds. This needs to be considered around daylight savings time boundaries.

Default: 0

producing_operation -- *a Operation field of model MULTIPLE*

The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field_Error for this field to be [unspecified], the default, if any Buffer_Plan needs to issue a producing Operation_Plan.

Default: [unspecified]

supplying_operation -- *a Operation field of model MULTIPLE*

Obsolete! This field has been renamed 'producing_operation' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.

Default: [unspecified]

Properties: obsolete=True

MULTIPLE_FILTER_AND_RANK -- *a flow_policy extension of model Buffer*

A MULTIPLE_Buffer issues replenishment Operations whose sizes are integral multiples of multiple_quantity and are bounded by quantity_range. The Operations are planned to arrive such that the Buffer's on_hand balance never drops below min_on_hand.

Up to 'excess_on_hand' more than the minimum required is allowed in the Buffer. By setting 'excess_on_hand' larger than zero, planning instability is reduced, and planning speed is increased. However, keeping it smaller minimizes excess inventory.

The MULTIPLE_FILTER_AND_RANK model has fields that references these models :

Operation.

min_on_hand -- *a Quantity field of model MULTIPLE_FILTER_AND_RANK*

The minimum on_hand balance that should be present in this Buffer. This Quantity is converted to the unit of this Buffer.

Default: 0

excess_on_hand -- *a Quantity field of model MULTIPLE_FILTER_AND_RANK*

Up to this Quantity more than the minimum required is allowed in this Buffer. Setting this above zero reduces instability, and speeds planning. Setting it smaller decreases excess inventory levels. This Quantity is converted to the unit of this Buffer.

Default: 0

quantity_range -- *a Quantity_Range field of model*

MULTIPLE_FILTER_AND_RANK

The range between which supplying operations of the buffer are constrained to lie.

Note that the smallest legal size of a supplying operation can be more than the min of the quantity_range, since it has to be a multiple of multiple_quantity. Similarly, the largest legal size can be smaller than the max of the quantity_range.

Default: [1, oo]

multiple_quantity -- *a Quantity field of model*

MULTIPLE_FILTER_AND_RANK

The Quantity of which the Quantity of the Item to be produced by each

producing_operation that is issued by this MULTIPLE_Buffer is a multiple of. This Quantity is converted to the unit of this Buffer. If multiple_quantity is set to '0', it implies that the quantities of producing operations can be any number (integer or non-integer) in the range of 'quantity_range'. The default is '0'.

Default: 0

min_time -- *a Time field of model MULTIPLE_FILTER_AND_RANK*

Items are planned such that they arrive at least min_time earlier than otherwise needed. This may be used as a safety time -- the next min_time of consumption will be maintained in the Buffer. min_time is represented internally as seconds so it specified as days, weeks, years, its will be converted to seconds. This needs to be considered around daylight savings time boundaries.

Default: 0

producing_operation -- *a Operation field of model*

MULTIPLE_FILTER_AND_RANK

The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field_Error for this field to be [unspecified], the default, if any Buffer_Plan needs to issue a producing Operation_Plan.

Default: [unspecified]

supplying operation -- a Operation field of model

MULTIPLE_FILTER_AND_RANK

Obsolete! This field has been renamed 'producing_operation' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.

Default: [unspecified]

Properties: obsolete=True

flow_plan_selection_start -- a Expression field of model

MULTIPLE_FILTER_AND_RANK

This expression is called before Flow_Plan are selected in order to build bags. For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to Buffer_Plan of this Buffer; problem_period : Bound to the problem period of the currently selected problem; problem_category : Bound to Problem_Category of the currently selected problem; active_strategy : Bound to Active_Strategy solving problem on this Buffer.

Example Usage:

Default: [unspecified]

flow_plan_selection_end -- a Expression field of model

MULTIPLE_FILTER_AND_RANK

This expression is called after Flow_Plan are selected to build the bags. For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to Buffer_Plan of this Buffer; problem_period : Bound to the problem period of the currently selected problem; problem_category : Bound to Problem_Category of the currently selected problem; active_strategy : Bound to Active_Strategy solving problem on this Buffer.

Example Usage:

Default: [unspecified]

flow_plan_selection_interval -- a Expression field of model

MULTIPLE_FILTER_AND_RANK

This expression returns the Date_Range over which resolver will look to select Flow_Plan in order to build bags. For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to Buffer_Plan of this Buffer; problem_period : Bound to the problem period of the currently selected problem; problem_category : Bound to Problem_Category of the currently selected problem; change_category : Bound to Strategy_Change_category for which resolver is building bag; active_strategy : Bound to Active_Strategy solving problem on this Buffer.

Example Usage:

Buffer b1 = supply_chains.find('Sup1').sites.find('Site1').buffers.find(Bu1)
b1.set_flow_plan_selection_interval('Date_Range(problem.start - 1 Week,
problem_end)')

Default: problem_period

flow_plan_filter -- a Expression field of model

MULTIPLE_FILTER_AND_RANK

This expression takes following parameters and if it returns true, the Flow_Plan will be selected and put into the bag. If it returns false, the Flow_Plan will not be put into the bag.

For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to Buffer_Plan of this Buffer; problem_period : Bound to the problem period of the currently selected problem; problem_category : Bound to Problem_Category of the currently selected problem; change_category : Bound to Strategy_Change_category for which resolver is building bag; active_strategy : Bound to Active_Strategy solving problem on this Buffer; flow_plan : Bound to Flow_Plan while doing bag candidate selection.

Example Usage:

Default: true

flow_plan_rank -- a Expression field of model

MULTIPLE_FILTER_AND_RANK

This expression returns the rank of the Flow_Plan which is used by the resolver in selecting the Flow_Plan. Higher rank means that the Flow_Plan is more likely to be chosen later from the bag.

For the convenience of OIL expression writer following special variables are available,

Extensions	MULTIPLE_FILTER_AND_RANK Extension
------------	------------------------------------

buffer_plan: : Bound to Buffer_Plan of this Buffer.problem.period : Bound to the problem.period of the currently selected problem.problem.category : Bound to Problem_Category of the currently selected problem.change.category : Bound to Strategy_Change.category for which resolver is building bag.active.strategy : Bound to Active_Strategy solving problem on this Buffer.flow_plan : Bound to Flow_Plan while doing bag candidate selection.

Example Usage:
Default: 1.0

continue_flow_plan_selection - - a Expression field of model
MULTIPLE_FILTER_AND_RANK

This expression takes following parameters and if it returns true, the selection of Flow_Plan will continue to be put into the bag. If it returns false, further Flow_Plans will not be selected.

For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to Buffer_Plan of this Buffer.problem.period : Bound to the problem.period of the currently selected problem.problem.category : Bound to Problem_Category of the currently selected problem.change.category : Bound to Strategy_Change.category for which resolver is building bag.active.strategy : Bound to Active_Strategy solving problem on this Buffer.flow_plan : Bound to Flow_Plan while doing bag candidate selection.

Example Usage:
Default: true

flow_plan_resize_quantity_range - - a Expression field of model
MULTIPLE_FILTER_AND_RANK
Replan restrictions This expression returns the Quantity_Range within which the selected Flow_Plan can be resized by the resolver.

For the convenience of OIL expression writer following special variables are available,

Extensions	MULTIPLE_FILTER_AND_RANK Extension
------------	------------------------------------

buffer_plan: : Bound to Buffer_Plan of this Buffer.problem.period : Bound to the problem.period of the currently selected problem.problem.category : Bound to Problem_Category of the currently selected problem.change.category : Bound to Strategy_Change.category for which resolver is building bag.active.strategy : Bound to Active_Strategy solving problem on this Buffer.flow_plan : Bound to Flow_Plan while doing bag candidate selection.

Example Usage:
Default: [unspecified]

flow_plan_move_restriction - - a Expression field of model
MULTIPLE_FILTER_AND_RANK

This expression returns the Date_Range within which the selected Flow_Plan can be moved by the resolver.

For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to Buffer_Plan of this Buffer.problem.period : Bound to the problem.period of the currently selected problem.problem.category : Bound to Problem_Category of the currently selected problem.change.category : Bound to Strategy_Change.category for which resolver is building bag.active.strategy : Bound to Active_Strategy solving problem on this Buffer.flow_plan : Bound to Flow_Plan while doing bag candidate selection.

Example Usage:
Default: [unspecified]

flow_plan_split_restriction - - a Expression field of model
MULTIPLE_FILTER_AND_RANK
This expression returns the Quantity_Range within which the selected Flow_Plan can be split by the resolver for a move

For the convenience of OIL expression writer following special variables are available,

buffer_plan: : Bound to Buffer_Plan of this Buffer.problem.period : Bound to the problem.period of the currently selected problem.problem.category : Bound to Problem_Category of the currently selected problem.change.category : Bound to Strategy_Change.category for which resolver is building bag.active.strategy : Bound to Active_Strategy solving problem on this Buffer.flow_plan : Bound to Flow_Plan while doing bag candidate selection.

Extensions	FIXED_QUANTITY_FENCED Extension
------------	---------------------------------

Example Usage:
Default: [unspecified]

FIXED_QUANTITY_FENCED -- *a flow_policy extension of model Buffer*

A **FIXED_QUANTITY_FENCED** Buffer issues replenishment Operations all with the same quantity. The Operations are planned to arrive such that the Buffer's on_hand balance never drops below 'min_on_hand'. This policy is often called "Fixed Order Quantity (FOQ)".

This is the same as **FIXED_QUANTITY** except that it has a 'fence' 'Horizon_Date' field which computes a Date at and after which a different fixed Quantity 'after_fence_quantity' is used, and a different 'after_fence_excess_on_hand'. This is used for more rough-cut planning a certain amount of time out in the horizon.

Up to 'excess_on_hand' more than the minimum required is allowed in the Buffer. By setting 'excess_on_hand' larger than zero, planning instability is reduced, and planning speed is increased. However, keeping it smaller minimizes excess inventory.

The **FIXED_QUANTITY_FENCED** model has fields that references these models:
Operation.

min_on_hand -- *a Quantity field of model FIXED_QUANTITY_FENCED*

The minimum on_hand balance that should be present in this Buffer. This Quantity is converted to the unit of this Buffer.

Default: 0

min_time -- *a Time field of model FIXED_QUANTITY_FENCED*

Items are planned such that they arrive at least 'min_time' earlier than otherwise needed. This may be used as a safety time -- the next 'min_time' of consumption will be maintained in the Buffer. 'min_time' is represented internally as seconds so if specified as as days, weeks, years, its will be converted to seconds. This needs to be considered around daylight savings time boundaries.

Default: 0

excess_on_hand -- *a Quantity field of model FIXED_QUANTITY_FENCED*

Up to this Quantity more than the minimum required is allowed in this Buffer. Setting this above zero reduces instability, and speeds planning. Setting it smaller decreases excess inventory levels. This Quantity is converted to the unit of this Buffer.

Default: 0

Extensions	FIXED_QUANTITY_FENCED Extension
------------	---------------------------------

quantity -- *a Quantity field of model FIXED_QUANTITY_FENCED*
The fixed Quantity of the Item to be produced by each 'producing_operation' planned before and not including 'fence' by this **FIXED_QUANTITY_FENCED** Buffer. This Quantity is converted to the preferred 'measure' of this Buffer. The default is "1", which means 1 unit of the Buffer.

Default: 1

fence -- *a Horizon_Date field of model FIXED_QUANTITY_FENCED*

The horizon-relative Date at and after which this flow_policy switches to coarser planning estimates. At and after the fence Date it begins using the 'after_fence_quantity' instead of 'quantity' and 'after_fence_excess_on_hand' instead of 'excess_on_hand'.
Default: oo

after_fence_excess_on_hand -- *a Quantity field of model FIXED_QUANTITY_FENCED*

Up to this Quantity more than the minimum required is allowed in this Buffer. Setting this above zero reduces instability, and speeds planning. Setting it smaller decreases excess inventory levels. This Quantity is converted to the unit of this Buffer.
Default: 0

after_fence_quantity -- *a Quantity field of model FIXED_QUANTITY_FENCED*

The fixed Quantity of the Item to be produced by each 'producing_operation' planned at and after 'fence' by this **FIXED_QUANTITY_FENCED** Buffer. This Quantity is converted to the preferred 'measure' of this Buffer. The default is "1", which means 1 unit of the Buffer.
Default: 1

producing_operation -- *a Operation field of model FIXED_QUANTITY_FENCED*

The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a field_error for this field to be [unspecified], the default, if any Buffer_Plan needs to issue a producing Operation_Plan.

Default: [unspecified]

supplying_operation -- *a Operation field of model FIXED_QUANTITY_FENCED*

Obsolete! This field has been renamed 'producing_operation' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.
Default: [unspecified]

Extensions	FIXED_TIME Extension
------------	----------------------

Properties: obsolete=True

FIXED_TIME -- a flow_policy extension of model Buffer

A FIXED_TIME Buffer is replenished with enough to go at least 'produce_time' without needing another producing operation. Producing operations are planned such that the Buffer's on_hand balance never drops below 'min_on_hand'. This policy is often called "Periodic Order Quantity (POQ)" or "Fixed Period Ordering".

Up to 'excess_on_hand' more than the minimum required is allowed in the Buffer. By setting 'excess_on_hand' larger than zero, planning instability is reduced, and planning speed is increased. However, keeping it smaller minimizes excess inventory.

The FIXED_TIME model has fields that references these models :
Operation.

produce_time -- a Time field of model FIXED_TIME

The fixed minimum Time desired between 'producing_operation's by this FIXED_TIME Buffer. Each 'producing_operation' is issued for enough to go at least 'produce_time' without dropping below 'min_on_hand'. 'produce_time' is represented internally as seconds so if specified as days, weeks, years, its will be converted to seconds. This needs to be considered around daylight savings time boundaries.

Default: 0

min_on_hand -- a Quantity field of model FIXED_TIME

The minimum on_hand balance that should be present in this Buffer. This Quantity is converted to the unit of this Buffer.

Default: 0

min_time -- a Time field of model FIXED_TIME

Items are planned such that they arrive at least 'min_time' earlier than otherwise needed. This may be used as a safety time -- the next 'min_time' of consumption will be maintained in the Buffer. 'min_time' is represented internally as seconds so if specified as days, weeks, years, its will be converted to seconds. This needs to be considered around daylight savings time boundaries.

Default: 0

excess_on_hand -- a Quantity field of model FIXED_TIME

Up to this Quantity more than the minimum required is allowed in this Buffer. Setting this above zero reduces instability, and speeds planning. Setting it smaller decreases excess inventory levels. This Quantity is converted to the unit of this Buffer.

Default: 0

Extensions	LFL_SIMPLE Extension
------------	----------------------

producing_operation -- a Operation field of model FIXED_TIME
The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field_Error for this field to be [unspecified], the default, if any Buffer_Plan needs to issue a producing Operation_Plan.

Default: [unspecified]

supplying_operation -- a Operation field of model FIXED_TIME

Obsolete! This field has been renamed 'producing_operation' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.

Default: [unspecified]

Properties: obsolete=True

supply_time -- a Time field of model FIXED_TIME

Obsolete! This field has been renamed 'produce_time' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.

Default: 0

Properties: obsolete=True

LFL_SIMPLE -- a flow_policy extension of model Buffer

A LFL_SIMPLE Buffer uses a simple lot-for-lot order policy: for each Operation that consumes from the Buffer, a 'producing_operation' with the same Quantity is issued upstream to fill the Buffer.

Excess material is never planned, but if excess material does appear with a planned 'producing_operation', the downstream Operation will be resized to match.

If an additional Operation_Plan produces into the Buffer, then a 'consuming_operation' (if not [unspecified]) will be generated to consume that excess material. If the 'consuming_operation' is [unspecified], then the excess will remain as on_hand until a downstream Operation consumes that amount. In that way the '1:1' upstream:downstream is maintained.

The LFL_SIMPLE model has fields that references these models :
Operation.

producing_operation - - *a Operation field of model LFL_SIMPLE*
The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field Error for this field to be [unspecified], the default, if any Buffer_Plan needs to issue a producing Operation_Plan.
Default: [unspecified]

consuming_operation - - *a Operation field of model LFL_SIMPLE*
The Operation to be created in order to consume additional flow from this Buffer as needed due to additional producing Operation_Plans.
Default: [unspecified]

supplying_operation - - *a Operation field of model LFL_SIMPLE*
Obsolete! This field has been renamed 'producing_operation' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.

Default: [unspecified]
Properties: obsolete=True

LFL_BOUNDED -- a flow_policy extension of model Buffer

A LFL_BOUNDED (Loi For Loi, Bounded) Buffer issues a single 'producing_operation' upstream, bounded in size by 'quantity_range', in order to satisfy each downstream Operation that consumes from the Buffer.

If the downstream Operation consumes more than 'quantity_range.max', then it is resized to 'quantity_range.max'. If the downstream Operation consumes less than 'quantity_range.min', the downstream Operation will be resized to 'quantity_range.min'.

If an upstream Operation_Plan produces excess material, the downstream Operation_Plan will be resized to match it. If an additional Operation_Plan or if excess material otherwise appears in the Buffer, then it will be used to reduce the quantity of one or more 'producing_operation's. No on_hand is intentionally maintained.

If an additional Operation_Plan produces into the Buffer, then a 'consuming_operation' (if not [unspecified]) will be generated to consume that excess material. If the 'consuming_operation' is [unspecified], then the excess will remain as on_hand until a downstream Operation consumes that amount. In that way, the 'upstream:downstream' relationship is maintained.

At and after the 'rough_fence' Horizon_Date, this flow_policy uses 'rough_quantity_range' rather than 'quantity_range'. Generally 'rough_quantity_range' will be somewhat larger, reducing the computational detail in more distant plans.

If 'rough_quantity_range' is from 0 to infinite (the default), then this flow_policy behaves like LFL_SIMPLE at and after 'rough_fence'. If 'quantity_range' is also from 0 to infinite, then this flow_policy behaves completely like LFL_SIMPLE.

If 'quantity_range' is set to a single value (min == max), then all producing and consuming Operation_Plans will be resized to that value.

The LFL_BOUNDED model has fields that reference these models :
Operation.

quantity_range - - *a Quantity_Range field of model LFL_BOUNDED*

The quantity_range of items of this Buffer for which to create a 'producing_operation'. If the consuming lot is for a Quantity outside this range, it will be resized to fit within this range.

This Quantity is converted to the unit of this Buffer.

If set to infinite, the default, this flow_policy behaves like LFL_SIMPLE.
Default: [0, oo]

rough_fence - - *a Horizon_Date field of model LFL_BOUNDED*

At and after this Horizon_Date, 'rough_quantity_range' is used rather than 'quantity_range'. Generally 'rough_quantity_range' will be a somewhat larger range, reducing the computational detail of more distant plans.
Default: infinite future

rough_quantity_range - - *a Quantity_Range field of model LFL_BOUNDED*
The quantity_range of items of this Buffer for which to create a 'producing_operation' at and after the 'rough_fence'. See 'quantity_range' for more detail. Generally, this value is a larger range than 'quantity_range', reducing the computational detail of more distant plans.

If set from 0 to infinite, the default, this flow_policy behaves like LFL_SIMPLE at and after the 'rough_fence'.

Extensions	MLFL_BOUNDED Extension
------------	------------------------

Default: [0, 00]

producing_operation - - *a Operation field of model LFL_BOUNDED*
The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field Error for this field to be [unspecified], the default, if any Buffer_Plan needs to issue a producing Operation_Plan.
Default: [unspecified]

consuming_operation - - *a Operation field of model LFL_BOUNDED*
The Operation to be created in order to consume additional flow from this Buffer as needed due to additional producing Operation_Plans.
Default: [unspecified]

supplying_operation - - *a Operation field of model LFL_BOUNDED*
Obsolete! This field has been renamed **producing_operation** in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.

Default: [unspecified]
Properties: obsolete=True

MLFL_BOUNDED - - *a flow_policy extension of model Buffer*

A MLFL_BOUNDED (Many-Lots For Lot, Bounded) Buffer issues one or more producing_operation's upstream, bounded in size by 'quantity_range', in order to satisfy each downstream Operation that consumes from the Buffer.

If the downstream Operation consumes more than 'quantity_range_max', then multiple producing_operation's will be needed. In such cases, the flow_policy attempts to keep the number of Operations as small as possible, and their Quantities relatively even.

For example, if 'quantity_range_max' is 100, and a downstream Operation consumes 240, three producing_operation's each with Quantity 80 is ideal. However, if the Quantity is later adjusted to 250, only one of the upstream orders will be adjusted to 90 (not each to 83, 83, and 84) in order to minimize volatility. Another increase from 250 to 260 will likely increase one of the other orders from 80 to 90 to keep the Quantities reasonably even.

If the downstream Operation consumes less than 'quantity_range_min', the downstream Operation will be resized to 'quantity_range_min'. Similarly, if excess material is produced by a producing_operation, then the downstream Operation will be resized.

Extensions	MLFL_BOUNDED Extension
------------	------------------------

If an additional Operation_Plan produces into the Buffer, then a consuming_operation (if not [unspecified]) will be generated to consume that excess material. If the consuming_operation is [unspecified], then one of the downstream operations (which is not defined) will be resized to consume the excess. In that way the N:1 upstream:downstream relationship is maintained, and no on_hand balance is planned.

At and after the 'rough_fence' Horizon_Date, this flow_policy uses 'rough_quantity_range' rather than 'quantity_range'. Generally 'rough_quantity_range' will be somewhat larger, reducing the computational detail in more distant plans.

If 'rough_quantity_range' is from 0 to infinite (the default), then this flow_policy behaves like LFL_SIMPLE at and after 'rough_fence'. If 'quantity_range' is also from 0 to infinite, then this flow_policy behaves completely like LFL_SIMPLE.

The MLFL_BOUNDED model has fields that references these models :
Operation.

quantity_range - - *a Quantity_Range field of model MLFL_BOUNDED*
The quantity range of items of this Buffer for which to create a producing_operation.

If the consuming lot is for a Quantity greater than this then the lots produced by the producing_operation will be broken into the largest balanced orders that are less than or equal to this Quantity. For example, if this is 100, and an order for 240 comes in, 3 orders will be issued, each with Quantity 80.

If the consuming lot is for a Quantity less than this then it will be resized

This Quantity is converted to the 'unit' of this Buffer.

If set to infinite, the default, this flow_policy behaves like LFL_SIMPLE.

Default: [0, 00]

rough_fence - - *a Horizon_Date field of model MLFL_BOUNDED*

At and after this Horizon_Date, 'rough_quantity_range' is used rather than 'quantity_range'. Generally 'rough_quantity_range' will be a somewhat larger range, reducing the computational detail of more distant plans.

Default: infinite future

Extensions	stocking_policy_extensions of model Buffer
------------	--

rough_quantity_range - - *a Quantity_Range field of model MLFL_BOUNDED*
The quantity range of items of this Buffer for which to create a 'producing_operation' at and after the 'rough_fence'. See 'quantity_range' for more detail. Generally, this value is a larger range than 'quantity_range', reducing the computational detail of more distant plans.

If set from 0 to infinite, the default, this flow_policy behaves like LFL_SIMPLE at and after the 'rough_fence'.
Default: [0, oo]

producing_operation - - *a Operation field of model MLFL_BOUNDED*
The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field_Error for this field to be [unspecified], the default, if any Buffer_Plan needs to issue a producing Operation_Plan.
Default: [unspecified]

consuming_operation - - *a Operation field of model MLFL_BOUNDED*
The Operation to be created in order to consume additional flow from this Buffer as needed due to additional producing Operation_Plan.
Default: [unspecified]

supplying_operation - - *a Operation field of model MLFL_BOUNDED*
Obsolete! This field has been renamed 'producing_operation' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.
Default: [unspecified]
Properties: obsolete=True

10.16.2 stocking_policy_extensions of model Buffer

MANUAL - - *a stocking_policy_extension of model Buffer*

A Manual stocking_policy implements no relevant stocking constraints. The user specifies the min_on_hand and min_time (in numbers but not a formula) in the flow_policy.

CALENDAR - - *a stocking_policy_extension of model Buffer*

Extensions	CALENDAR Extension
------------	--------------------

A CALENDAR stocking_policy computes safety stock (min_on_hand and min_time) based on mean and standard deviation of lead time and demand. Other inputs are the replenishment interval or quantity. It computes safety stock (min_on_hand and min_time) one buffer at a time i.e. using local information. It is assumed that the demand on this buffer is normally distributed.

Currently CALENDAR stocking_policy is only implemented for the PRODUCING_FLOW_CALENDAR flow_policy. Note that the user can either bias or override the automatically computed safety stock by specifying a custom formula. See the user_bias field description for details.

Most of the fields in the 'CALENDAR' stocking_policy can be specified in a Calendar, so they can change over time.

The CALENDAR model has fields that references these models :
Calendar.

units_of_safety_stock - - *a Safety_Stock_Units field of model CALENDAR*
Specifies if min_on_hand and/or min_time should be automatically computed.

If QUANTITY then the safety stock will be calculated in quantity of units and only min_on_hand will be automatically set by the stocking_policy.

If TIME then the safety stock will be calculated in days of coverage and only min_time will be automatically set by the stocking_policy.

If QUANTITY_TIME then the safety stock will be calculated in both quantity of units and days of coverage. First min_on_hand will be computed and then converted to min_time.
Default: QUANTITY

user_bias - - *a Expression field of model CALENDAR*
Used for either biasing or overriding the automatically computed safety stock by specifying a custom formula in terms of an OIL expression. If min_on_hand and min_time are both automatically calculated then the bias is first applied to min_on_hand and then the biased min_on_hand is converted into min_time.

When the expression executes:
'buffer' is set to the buffer whose stocking policy is being calculated.
'min_on_hand' is set to the min_on_hand calculated
'start_date' is the starting date where the min_on_hand applies
'end_date' is the ending date where the min_on_hand applies

Extensions	CALENDAR Extension
------------	--------------------

mean_lead_time' A Time
 std_dev_lead_time' A Time
 mean_demand' A Quantity
 std_dev_demand' A Quantity
 The result from the expression will be used as the min_on_hand for the buffer's
 flow_policy from start_date to end_date.

Examples:

1. Additive Bias:

Setting user_bias to min_on_hand + 10' will first compute min_on_hand using the
 pre-defined formula and then add 10 to that number.

2. Multiplicative Bias:

Setting user_bias to min_on_hand * 1.2' will first compute min_on_hand using the
 pre-defined formula and then increase it 1.2 fold.

3. Time-phased Bias:

Setting user_bias to if (and (start_date >= "1/1/97", end_date < "4/1/97"),
 min_on_hand * 1.2, min_on_hand * 1.1)' will first compute min_on_hand using the
 pre-defined formula and then increase it 1.2 fold for the first quarter of 1997 while
 increase it 1.1 fold for other times.

4. Overriding Pre-defined Formula:

Setting user_bias to mean_demand * 0.25' sets the min_on_hand to the 1/4th of the
 mean_demand. Note that the safety stock computation due to the pre-defined formula
 is by-passed here with a user-specified formula.

Default: min_on_hand

customer_service_level - - a Percentage field of model CALENDAR

The expected level of service from this buffer. This field can not be set to 100 as that
 would result in infinite safety stock.

Default: 99

default_mean_demand - - a Quantity field of model CALENDAR

Mean demand is the average demand rate (in units/days) put on this buffer by buffers
 immediately downstream to it.

Extensions	CALENDAR Extension
------------	--------------------

Default: 0

default_mean_demand_time_bucket - - a Time field of model CALENDAR

Specifies the time bucket over which mean_demand is defined.

Default: 1 Day

mean_demand_calendar - - a Calendar field of model CALENDAR

Time-phased average demand rate (in units/days).

Default: [unspecified]

default_standard_deviation_demand - - a Quantity field of model CALENDAR

Standard deviation of demand is the variation in demand put on this buffer by buffers
 immediately downstream to it.

Default: 0

standard_deviation_demand_calendar - - a Calendar field of model CALENDAR

Time-phased standard deviation of demand.

Default: [unspecified]

default_mean_lead_time - - a Time field of model CALENDAR

Mean lead time is the average time to produce/supply material into this buffer from
 buffers immediately upstream to it.

Default: 0

mean_lead_time_calendar - - a Calendar field of model CALENDAR

Time-phased mean lead time.

Default: [unspecified]

default_standard_deviation_lead_time - - a Time field of model CALENDAR

Standard deviation of lead time is the variation in producing/supplying material into
 this buffer from buffers immediately upstream to it.

Default: 0

standard_deviation_lead_time_calendar - - a Calendar field of model CALENDAR

Default: [unspecified]

Time-phased standard deviation of lead time.

Default: [unspecified]

safety_factor, safety_factor (Date) - - a Quantity field of model CALENDAR

Safety factor scales the supply and demand variations in accordance to the customer
 service level expectations.

Extensions	CALENDAR Extension
------------	--------------------

Properties: Export-Only Field

demand_safety_stock, demand_safety_stock (Date) -- *a* Quantity field of model
CALENDAR
Safety stock due to demand variation only.
Properties: Export-Only Field

supply_safety_stock, supply_safety_stock (Date) -- *a* Quantity field of model
CALENDAR
Safety stock due to supply variation only.
Properties: Export-Only Field

Extensions	Buffer_Problem_Detector Extensions
------------	------------------------------------

10.17 Buffer_Problem_Detector Extensions

10.17.1 detector extensions of model Buffer_Problem_Detector

10.18 Buffer_Plan Extensions

10.18.1 Flow_policy extensions of model Buffer_Plan

BUCKETED_NESTED_SORT -- a flow_policy extension of model Buffer_Plan

A BUCKETED_NESTED_SORT Buffer manages the Flow_Plan in buckets of time, as specified by the horizon. See the corresponding 'flow_policy' of the Buffer model for more information.

The BUCKETED_NESTED_SORT model has these submodels:

Sorted_Bucket.

The BUCKETED_NESTED_SORT model has fields that reference these models:

Sorted_Bucket.

sorted_buckets -- a list of Sorted_Bucket submodels of model

BUCKETED_NESTED_SORT

allocate_excess -- a Void field of model BUCKETED_NESTED_SORT

This command allocates unused material in the Buffer_Plan by creating forecast Requests for the default_product root of the Buffer.

Properties: command=True Export-Only Field

PRODUCING_FLOW_CALENDAR -- a flow_policy extension of model Buffer_Plan

A PRODUCING_FLOW_CALENDAR Buffer maintains a min_time safety time and a min_on_hand safety quantity. See the corresponding 'flow_policy' of the Buffer model for more details.

max_target_profile, max_target_profile (Date_Range) -- a

List(Profile_Quantity) field of model PRODUCING_FLOW_CALENDAR

A profile or a list of changes to max allowable Quantity's during the specified finite Date_Range. Up to max_target Quantity is allowed in this buffer. If the

Buffer_Plan on_hand Quantity goes above Buffer_Plan.max_target Quantity,

RHYTHM will detect EXCESS_ON_HAND problem.

Properties: command=True Export-Only Field

max_target (Date), max_target (Date_Range, Logical) -- a Quantity field of model PRODUCING_FLOW_CALENDAR

Maximum target Quantity allowed in this BASIC buffer. If the Buffer_Plan on_hand Quantity goes above Buffer_Plan.max_target Quantity, RHYTHM will detect EXCESS_ON_HAND problem.

If a Date is passed, this returns the max_target Quantity allowed as of that Date in this buffer.

If a Date_Range is passed, this returns the smallest max_target Quantity allowed during that Date_Range in this buffer.

If a Date_Range and a Logical is passed, this returns the smallest max_target Quantity allowed during that Date_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest max_target Quantity allowed during that Date_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

safety_target_profile, safety_target_profile (Date_Range) -- a

List(Profile_Quantity) field of model PRODUCING_FLOW_CALENDAR

A profile or a list of changes to the safety_target Quantity during the specified Date_Range. This safety_target Quantity is the max of Buffer.min_on_hand and the Quantity required to cover the next min_time of consumption in this Buffer.

For example, consider a Buffer with min_on_hand is set to "100", min_time is set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the safety_target_profile will have three entries, first entry with quantity 100 until 4/1, second entry with quantity 200 starting on 4/1 and third entry with quantity 100 starting on 4/15. In the second period "200" will be maintained in order to cover the next "2 weeks" of out flows.

Properties: command=True Export-Only Field

safety_target (Date), safety_target (Date_Range, Logical) -- a Quantity field of model PRODUCING_FLOW_CALENDAR

Safety target is a minimum Quantity this BASIC Buffer is required to maintain based on existing plan. Mathematically, safety_target Quantity is the max of Buffer.min_on_hand and the Quantity required to cover the next Buffer.min_time of consumption in this Buffer.

For example, consider a Buffer with min_on_hand set to "100" and min_time set to "2 weeks". If there is currently just one unit of flow planned from the Buffer, and it is for 200 units on 4/15, then the safety_target Quantity on date 4/10 will be 200.

If a Date is passed, this returns the safety_target Quantity required to be maintained as of that Date considering the existing plan.

If a Date_Range is passed, this returns the smallest safety_target Quantity required to be maintained during that Date_Range considering the existing plan.

If a Date_Range and a Logical is passed, this returns the smallest safety_target Quantity required to be maintained during that Date_Range considering the existing plan if the Logical is True. If the Logical is False, this returns the largest safety_target Quantity required to be maintained during that Date_Range.

The Quantity is converted to the unit of this buffer.
Properties: Export-Only Field

cycle_on_hand_profile, cycle_on_hand (Date_Range) -- a
List(Profile_Quantity) field of model PRODUCING_FLOW_CALENDAR
A profile or a list of changes to cycle_on_hand (max(0,0, Buffer_Plan_on_hand - Buffer_Plan_safety_target)) Quantity planned during the specified finite Date_Range in this buffer.
Properties: command=True Export-Only Field

cycle_on_hand (Date), cycle_on_hand (Date_Range, Logical) -- a Quantity
field of model PRODUCING_FLOW_CALENDAR
Cycle_on_hand (max(0,0, Buffer_Plan_on_hand - Buffer_Plan_safety_target)) Quantity planned in this buffer.

If a Date is passed, this returns the cycle_on_hand Quantity planned as of that Date in this buffer.

If a Date_Range is passed, this returns the smallest cycle_on_hand Quantity planned during that Date_Range in this buffer.

If a Date_Range and a Logical is passed, this returns the smallest cycle_on_hand Quantity planned during that Date_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest max_target Quantity planned during that Date_Range.

The Quantity is converted to the unit of this buffer.
Properties: Export-Only Field

excess_on_hand_profile, excess_on_hand_profile (Date_Range) -- a
List(Profile_Quantity) field of model PRODUCING_FLOW_CALENDAR
A profile or a list of changes to excess_on_hand Quantity during the specified finite Date_Range. This is the Buffer_Plan_on_hand Quantity above allowable Buffer_Plan_max_target Quantity. Mathematically, excess_on_hand is computed using formula, max(0,0, Buffer_Plan_on_hand - Buffer_Plan_max_target).
Properties: command=True Export-Only Field

excess_on_hand (Date), excess_on_hand (Date_Range, Logical) -- a Quantity
field of model PRODUCING_FLOW_CALENDAR
Excess_on_hand Quantity planned above the allowable Buffer_excess_on_hand (will be renamed to Buffer_max_cycle_quantity) Quantity in this buffer. Mathematically, excess_on_hand is computed using formula, max(0,0, Buffer_Plan_on_hand - Buffer_Plan_max_target).

If a Date is passed, this returns the excess_on_hand Quantity planned as of that Date.

If a Date_Range is passed, this returns the smallest excess_on_hand Quantity planned during that Date_Range.

If a Date_Range and a Logical is passed, this returns the smallest excess_on_hand Quantity planned during that Date_Range, if the Logical is True. If the Logical is False, this returns the largest excess_on_hand Quantity planned during that Date_Range.

The Quantity is converted to the unit of this buffer.
Properties: Export-Only Field

low_on_hand_profile, low_on_hand_profile (Date_Range) -- a
List(Profile_Quantity) field of model PRODUCING_FLOW_CALENDAR
A profile or a list of changes to the low_on_hand Quantity during the specified Date_Range. Mathematically, low_on_hand is computed using following formula, (Buffer_Plan_safety_target(0) - Buffer_Plan_on_hand(0)).
Properties: command=True Export-Only Field

low_on_hand (Date), **low_on_hand** (Date, Range, Logical) -- *a Quantity field of model PRODUCING_FLOW_CALENDAR*
It is the 'on_hand' Quantity planned below the minimum 'safety_target' Quantity. Mathematically, 'low_on_hand' is computed using following formula,
(Buffer_Plan.safety_target - Buffer_Plan.on_hand).

If a Date is passed, this returns the 'low_on_hand' Quantity planned as of that Date.

If a Date, Range is passed, this returns the smallest 'low_on_hand' Quantity planned during that Date, Range.

If a Date, Range and a Logical is passed, this returns the smallest 'low_on_hand' Quantity planned during that Date, Range, if the Logical is True. If the Logical is False, this returns the largest 'low_on_hand' Quantity planned during that Date, Range.

The Quantity is converted to the 'unit' of this Buffer.
Properties: Export-Only Field

PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK -- *a flow_policy extension of model Buffer_Plan*

Just like Producing Flow Calendar Flow Policy

max_target_profile, **max_target_profile** (Date, Range) -- *a List(Profile_Quantity) field of model*

PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK

A profile or a list of changes to max allowable Quantity's during the specified finite Date, Range. Up to 'max_target' Quantity is allowed in this buffer. If the Buffer_Plan.on_hand' Quantity goes above Buffer_Plan.max_target' Quantity, RHYTHM will detect EXCESS_ON_HAND problem.

Properties: command=True Export-Only Field

max_target (Date), **max_target** (Date, Range, Logical) -- *a Quantity field of model PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK*
Maximum target Quantity allowed in this BASIC buffer. If the Buffer_Plan.on_hand' Quantity goes above Buffer_Plan.max_target' Quantity, RHYTHM will detect EXCESS_ON_HAND problem.

If a Date is passed, this returns the 'max_target' Quantity allowed as of that Date in this buffer.

If a Date, Range is passed, this returns the smallest 'max_target' Quantity allowed during that Date, Range in this buffer.

If a Date, Range and a Logical is passed, this returns the smallest 'max_target' Quantity allowed during that Date, Range in this buffer. If the Logical is True, If the Logical is False, this returns the largest 'max_target' Quantity allowed during that Date, Range.

The Quantity is converted to the 'unit' of this buffer.

Properties: Export-Only Field

safety_target_profile, **safety_target_profile** (Date, Range) -- *a List(Profile_Quantity) field of model*

PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK

A profile or a list of changes to the 'safety_target' Quantity during the specified Date, Range. This 'safety_target' Quantity is the max of Buffer.min_on_hand and the Quantity required to cover the next 'min_time' of consumption in this Buffer.

For example, consider a Buffer with 'min_on_hand' set to "100", 'min_time' is set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the 'safety_target_profile' will have three entries, first entry with quantity 100 until 4/1, second entry with quantity 200 starting on 4/1 and third entry with quantity 100 starting on 4/15. In the second period "200" will be maintained in order to cover the next "2 weeks" of out flows.

Properties: command=True Export-Only Field

safety_target (Date), **safety_target** (Date, Range, Logical) -- *a Quantity field of model PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK*
Safety target is a minimum Quantity this BASIC Buffer is required to maintain based on existing plan. Mathematically, 'safety_target' Quantity is the max of Buffer.min_on_hand and the Quantity required to cover the next 'Buffer.min_time' of consumption in this Buffer.

For example, consider a Buffer with 'min_on_hand' set to "100" and 'min_time' set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the 'safety_target' Quantity on date 4/10 will be 200.

If a Date is passed, this returns the 'safety_target' Quantity required to be maintained as of that Date considering the existing plan.

If a Date, Range is passed, this returns the smallest 'safety_target' Quantity required to be maintained during that Date, Range considering the existing plan.

If a Date_Range and a Logical is passed, this returns the smallest 'safety_target' Quantity required to be maintained during that Date_Range considering the existing plan if the Logical is True. If the Logical is False, this returns the largest 'safety_target' Quantity required to be maintained during that Date_Range.

The Quantity is converted to the 'unit' of this buffer.

Properties: Export-Only Field

cycle_on_hand_profile, cycle_on_hand_profile (Date_Range) --a

List(Profile_Quantity) field of model

PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK

A profile or a list of changes to cycle_on_hand' (max(0.0, Buffer_Plan_on_hand - Buffer_Plan_safety_target)) Quantity's planned during the specified finite Date_Range in this buffer.

Properties: command=True Export-Only Field

cycle_on_hand (Date), cycle_on_hand (Date_Range, Logical) --a Quantity field of model PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK
Cycle_on_hand' (max(0.0, Buffer_Plan_on_hand - Buffer_Plan_safety_target)) Quantity planned in this buffer.

If a Date is passed, this returns the 'cycle_on_hand' Quantity planned as of that Date in this buffer.

If a Date_Range is passed, this returns the smallest 'cycle_on_hand' Quantity planned during that Date_Range in this buffer.

If a Date_Range and a Logical is passed, this returns the smallest 'cycle_on_hand' Quantity planned during that Date_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest 'max_target' Quantity planned during that Date_Range.

The Quantity is converted to the 'unit' of this buffer.

Properties: Export-Only Field

excess_on_hand_profile, excess_on_hand_profile (Date_Range) --a
List(Profile_Quantity) field of model

PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK

A profile or a list of changes to excess_on_hand' Quantity during the specified finite Date_Range. This is the 'Buffer_Plan_on_hand' Quantity above allowable Buffer_Plan_max_target Quantity. Mathematically, 'excess_on_hand' is computed using formula, max(0.0, Buffer_Plan_on_hand - Buffer_Plan_max_target).

Properties: command=True Export-Only Field

excess_on_hand (Date), excess_on_hand (Date_Range, Logical) --a Quantity field of model PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK
Excess_on_hand' Quantity planned above the allowable 'Buffer:excess_on_hand' will be renamed to Buffer:excess_on_hand' Quantity in this buffer. Mathematically, 'excess_on_hand' is computed using formula, max(0.0, Buffer_Plan_on_hand - Buffer_Plan_max_target).

If a Date is passed, this returns the 'excess_on_hand' Quantity planned as of that Date.

If a Date_Range is passed, this returns the smallest 'excess_on_hand' Quantity planned during that Date_Range.

If a Date_Range and a Logical is passed, this returns the smallest 'excess_on_hand' Quantity planned during that Date_Range, if the Logical is True. If the Logical is False, this returns the largest 'excess_on_hand' Quantity planned during that Date_Range.

The Quantity is converted to the 'unit' of this buffer.

Properties: Export-Only Field

low_on_hand_profile, low_on_hand_profile (Date_Range) --a

List(Profile_Quantity) field of model

PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK

A profile or a list of changes to the 'low_on_hand' Quantity during the specified Date_Range. Mathematically, 'low_on_hand' is computed using following formula, (Buffer_Plan_safety_target(d) - Buffer_Plan_on_hand(d)).

Properties: command=True Export-Only Field

low_on_hand (Date), low_on_hand (Date_Range, Logical) --a Quantity field of model PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK
It is the 'on_hand' Quantity planned below the minimum 'safety_target' Quantity. Mathematically, 'low_on_hand' is computed using following formula, (Buffer_Plan_safety_target - Buffer_Plan_on_hand).

Extensions	ON_HAND_CALENDAR Extension
------------	----------------------------

If a Date is passed, this returns the `low_on_hand` Quantity planned as of that Date.

If a `Date_Range` is passed, this returns the smallest `low_on_hand` Quantity planned during that `Date_Range`.

If a `Date_Range` and a Logical is passed, this returns the smallest `low_on_hand` Quantity planned during that `Date_Range`. If the Logical is True. If the Logical is False, this returns the largest `low_on_hand` Quantity planned during that `Date_Range`.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

ON_HAND_CALENDAR -- a flow_policy extension of model Buffer_Plan

An `ON_HAND_CALENDAR` Buffer maintains a `min_time` safety time and a `min_on_hand` safety quantity. See the corresponding `flow_policy` of the Buffer model for more details.

`capacity (Date_Range) -- a Quantity field of model ON_HAND_CALENDAR`

This returns the gross capacity in the specified date range given the underlying `on_hand_calendar`. Note that in cases where the `date_range` does not completely enclose the `bucket_spec` of the calendar, the entire capacity of the bucket is available for consumption and is returned.

Properties: Export-Only Field

ON_HAND_CALENDAR_FILTER_AND_RANK -- a flow_policy extension of model Buffer_Plan

An `ON_HAND_CALENDAR` Buffer maintains a `min_time` safety time and a `min_on_hand` safety quantity. See the corresponding `flow_policy` of the Buffer model for more details.

`capacity (Date_Range) -- a Quantity field of model`

`ON_HAND_CALENDAR_FILTER_AND_RANK`

This returns the gross capacity in the specified date range given the underlying `on_hand_calendar`. Note that in cases where the `date_range` does not completely enclose the `bucket_spec` of the calendar, the entire capacity of the bucket is available for consumption and is returned.

Properties: Export-Only Field

Extensions	FLOW_LIMIT_CALENDAR Extension
------------	-------------------------------

FLOW_LIMIT_CALENDAR -- a flow_policy extension of model Buffer_Plan

A `FLOW_LIMIT_CALENDAR` Buffer has a behaviour identical to a `ON_HAND_CALENDAR` except that the `NEGATIVE_ON_HAND` problems are in this case called `OVER_FLOW_LIMIT` problems. See the corresponding `flow_policy` of the Buffer model for more details.

`capacity (Date_Range) -- a Quantity field of model FLOW_LIMIT_CALENDAR`

This returns the gross capacity in the specified date range given the underlying `flow_limit_calendar`. Note that in cases where the `date_range` does not completely enclose the `bucket_spec` of the calendar, the entire capacity of the bucket is available for consumption and is returned.

Properties: Export-Only Field

FLOW_LIMIT_CALENDAR_FILTER_AND_RANK -- a flow_policy extension of model Buffer_Plan

A `FLOW_LIMIT_CALENDAR` Buffer has a behaviour identical to a `ON_HAND_CALENDAR` except that the `NEGATIVE_ON_HAND` problems are in this case called `OVER_FLOW_LIMIT` problems. See the corresponding `flow_policy` of the Buffer model for more details.

`capacity (Date_Range) -- a Quantity field of model`

`FLOW_LIMIT_CALENDAR_FILTER_AND_RANK`

This returns the gross capacity in the specified date range given the underlying `flow_limit_calendar`. Note that in cases where the `date_range` does not completely enclose the `bucket_spec` of the calendar, the entire capacity of the bucket is available for consumption and is returned.

Properties: Export-Only Field

BASIC -- a flow_policy extension of model Buffer_Plan

A `BASIC` Buffer maintains a `min_time` safety time and a `min_on_hand` safety quantity. See the corresponding `flow_policy` of the Buffer model for more details.

`max_target_profile, max_target_profile (Date_Range) -- a`

`List(Profile_Quantity) field of model BASIC`

A profile or a list of changes to max allowable Quantity's during the specified finite `Date_Range`. Up to `max_target` Quantity is allowed in this buffer. If the `Buffer_Plan.on_hand` Quantity goes above `Buffer_Plan.max_target` Quantity, `RHYTHM` will detect `EXCESS_ON_HAND` problem.

Properties: command=True Export-Only Field

`max_target (Date), max_target (Date_Range, Logical) -- a Quantity/field of model BASIC`
 Maximum target Quantity allowed in this BASIC buffer. If the Buffer_Plan_on_hand' Quantity goes above Buffer_Plan_max_target' Quantity, RHYTHM will detect EXCESS_ON_HAND problem.

If a Date is passed, this returns the 'max_target' Quantity allowed as of that Date in this buffer.

If a Date_Range is passed, this returns the smallest 'max_target' Quantity allowed during that Date_Range in this buffer.

If a Date_Range and a Logical is passed, this returns the smallest 'max_target' Quantity allowed during that Date_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest 'max_target' Quantity allowed during that Date_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

`safety_target_profile, safety_target_profile (Date_Range) -- a`

`List(Profile_Quantity)/field of model BASIC`

A profile or a list of changes to the 'safety_target' Quantity during the specified Date_Range. This 'safety_target' Quantity is the max of Buffer_min_on_hand and the Quantity required to cover the next 'min_time' of consumption in this Buffer.

For example, consider a Buffer with 'min_on_hand' is set to "100", 'min_time' is set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the 'safety_target_profile' will have three entries, first entry with quantity 100 until 4/1, second entry with quantity 200 starting on 4/1 and third entry with quantity 100 starting on 4/15. In the second period "200" will be maintained in order to cover the next "2 weeks" of out flows.

Properties: command=True Export-Only Field

`safety_target (Date), safety_target (Date_Range, Logical) -- a Quantity/field of model BASIC`

Safety target is a minimum Quantity this BASIC Buffer is required to maintain based on existing plan. Mathematically, 'safety_target' Quantity is the max of Buffer_min_on_hand and the Quantity required to cover the next 'Buffer_min_time' of consumption in this Buffer.

For example, consider a Buffer with 'min_on_hand' set to "100" and 'min_time' set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the 'safety_target' Quantity on date 4/10 will be 200.

If a Date is passed, this returns the 'safety_target' Quantity required to be maintained as of that Date considering the existing plan.

If a Date_Range is passed, this returns the smallest 'safety_target' Quantity required to be maintained during that Date_Range considering the existing plan.

If a Date_Range and a Logical is passed, this returns the smallest 'safety_target' Quantity required to be maintained during that Date_Range considering the existing plan if the Logical is True. If the Logical is False, this returns the largest 'safety_target' Quantity required to be maintained during that Date_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

`cycle_on_hand_profile, cycle_on_hand_profile (Date_Range) -- a`

`List(Profile_Quantity)/field of model BASIC`

A profile or a list of changes to 'cycle_on_hand' (max(0,0, Buffer_Plan_on_hand - Buffer_Plan_safety_target)) Quantity's planned during the specified finite Date_Range in this buffer.

Properties: command=True Export-Only Field

`cycle_on_hand (Date), cycle_on_hand (Date_Range, Logical) -- a Quantity/field of model BASIC`

Cycle 'on_hand' (max(0,0, Buffer_Plan_on_hand - Buffer_Plan_safety_target)) Quantity planned in this buffer.

If a Date is passed, this returns the 'cycle_on_hand' Quantity planned as of that Date in this buffer.

If a Date_Range is passed, this returns the smallest 'cycle_on_hand' Quantity planned during that Date_Range in this buffer.

If a Date_Range and a Logical is passed, this returns the smallest 'cycle_on_hand' Quantity planned during that Date_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest 'max_target' Quantity planned during that Date_Range.

excess_on_hand (Date), **excess_on_hand** (Date_Range, Logical) -- *a* Quantity *field of model* BASIC_FILTER_AND_RANK

Excess on_hand' Quantity planned above the allowable 'Buffer:excess_on_hand'(will be renamed to Buffer:excess_on_hand) Quantity in this buffer. Mathematically, excess_on_hand' is computed using formula, $\max(0, \text{Buffer_Plan.on_hand} - \text{Buffer_Plan.max_target})$.

If a Date is passed, this returns the excess_on_hand' Quantity planned as of that Date.

If a Date_Range is passed, this returns the smallest excess_on_hand' Quantity planned during that Date_Range.

If a Date_Range and a Logical is passed, this returns the smallest excess_on_hand' Quantity planned during that Date_Range, if the Logical is True. If the Logical is False, this returns the largest excess_on_hand' Quantity planned during that Date_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

low_on_hand_profile, **low_on_hand** (Date_Range) -- *a* List(Profile_Quantity) *field of model* BASIC_FILTER_AND_RANK

A profile or a list of changes to the low_on_hand' Quantity during the specified Date_Range. Mathematically, low_on_hand' is computed using following formula, (Buffer_Plan.safety_target() - Buffer_Plan.on_hand()).

Properties: command=True Export-Only Field

low_on_hand (Date), **low_on_hand** (Date_Range, Logical) -- *a* Quantity *field of model* BASIC_FILTER_AND_RANK

It is the low_on_hand' Quantity planned below the minimum safety_target' Quantity. Mathematically, low_on_hand' is computed using following formula, (Buffer_Plan.safety_target - Buffer_Plan.on_hand).

If a Date is passed, this returns the low_on_hand' Quantity planned as of that Date.

If a Date_Range is passed, this returns the smallest low_on_hand' Quantity planned during that Date_Range.

If a Date_Range and a Logical is passed, this returns the smallest low_on_hand' Quantity planned during that Date_Range, if the Logical is True. If the Logical is False, this returns the largest low_on_hand' Quantity planned during that Date_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

FIXED_QUANTITY -- *a* flow_policy extension of model Buffer_Plan

A FIXED_QUANTITY Buffer maintains a min_time' safety time and a min_on_hand' safety quantity. See the corresponding flow_policy' of the Buffer model for more details.

max_target_profile, **max_target** (Date_Range) -- *a* List(Profile_Quantity) *field of model* FIXED_QUANTITY

A profile or a list of changes to max allowable Quantity's during the specified finite Date_Range. Up to max_target' Quantity is allowed in this buffer. If the Buffer_Plan.on_hand' Quantity goes above Buffer_Plan.max_target' Quantity, RHYTHM will detect EXCESS_ON_HAND problem.

Properties: command=True Export-Only Field

max_target (Date), **max_target** (Date_Range, Logical) -- *a* Quantity *field of model* FIXED_QUANTITY

Maximum target Quantity allowed in this BASIC buffer. If the Buffer_Plan.on_hand' Quantity goes above 'Buffer_Plan.max_target' Quantity, RHYTHM will detect EXCESS_ON_HAND problem.

If a Date is passed, this returns the max_target' Quantity allowed as of that Date in this buffer.

If a Date_Range is passed, this returns the smallest max_target' Quantity allowed during that Date_Range in this buffer.

If a Date_Range and a Logical is passed, this returns the smallest max_target' Quantity allowed during that Date_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest max_target' Quantity allowed during that Date_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

safety_target_profile, safety_target_profile (Date, Range) -- a

List(Profile_Quantity) *field of model FIXED_QUANTITY*

A profile or a list of changes to the safety_target_Quantity during the specified Date_Range. This safety_target_Quantity is the max of Buffer_min_on_hand and the Quantity required to cover the next min_time of consumption in this Buffer.

For example, consider a Buffer with min_on_hand set to "100", min_time is set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the safety_target_profile will have three entries, first entry with quantity 100 until 4/1, second entry with quantity 200 starting on 4/1 and third entry with quantity 100 starting on 4/15. In the second period "200" will be maintained in order to cover the next "2 weeks" of out flows.

Properties: command=True Export-Only Field

safety_target (Date), safety_target (Date, Range, Logical) -- a Quantity field of model FIXED_QUANTITY

Safety_target is a minimum Quantity this BASIC Buffer is required to maintain based on existing Plan. Mathematically, safety_target_Quantity is the max of Buffer_min_on_hand and the Quantity required to cover the next Buffer_min_time of consumption in this Buffer.

For example, consider a Buffer with min_on_hand set to "100" and min_time set to "2 weeks". If there is currently just one flow planned from the Buffer, and it is for 200 units on 4/15, then the safety_target_Quantity on date 4/10 will be 200.

If a Date is passed, this returns the safety_target_Quantity required to be maintained as of that Date considering the existing plan.

If a Date_Range is passed, this returns the smallest safety_target_Quantity required to be maintained during that Date_Range considering the existing plan.

If a Date_Range and a Logical is passed, this returns the smallest safety_target_Quantity required to be maintained during that Date_Range considering the existing plan if the Logical is True. If the Logical is False, this returns the largest safety_target_Quantity required to be maintained during that Date_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

cycle_on_hand_profile, cycle_on_hand_profile (Date, Range) -- a

List(Profile_Quantity) *field of model FIXED_QUANTITY*

A profile or a list of changes to cycle_on_hand (max(0,0, Buffer_Plan_on_hand - Buffer_Plan_safety_target)) Quantity's planned during the specified finite Date_Range in this buffer.

Properties: command=True Export-Only Field

cycle_on_hand (Date), cycle_on_hand (Date, Range, Logical) -- a Quantity field of model FIXED_QUANTITY

Cycle_on_hand (max(0,0, Buffer_Plan_on_hand - Buffer_Plan_safety_target)) Quantity planned in this buffer.

If a Date is passed, this returns the cycle_on_hand_Quantity planned as of that Date in this buffer.

If a Date_Range is passed, this returns the smallest cycle_on_hand_Quantity planned during that Date_Range in this buffer.

If a Date_Range and a Logical is passed, this returns the smallest cycle_on_hand_Quantity planned during that Date_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest max_target_Quantity planned during that Date_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

excess_on_hand_profile, excess_on_hand_profile (Date, Range) -- a

List(Profile_Quantity) *field of model FIXED_QUANTITY*

A profile or a list of changes to excess_on_hand_Quantity during the specified finite Date_Range. This is the Buffer_Plan_on_hand_Quantity above allowable Buffer_Plan_max_target_Quantity. Mathematically, excess_on_hand is computed using formula, max(0,0, Buffer_Plan_on_hand - Buffer_Plan_max_target).

Properties: command=True Export-Only Field

excess_on_hand (Date), excess_on_hand (Date, Range, Logical) -- a Quantity field of model FIXED_QUANTITY

Excess_on_hand_Quantity planned above the allowable Buffer_excess_on_hand will be renamed to Buffer_max_cycle_quantity) Quantity in this buffer. Mathematically, excess_on_hand is computed using formula, max(0,0, Buffer_Plan_on_hand - Buffer_Plan_max_target).

Extensions	MULTIPLE Extension
------------	--------------------

If a Date is passed, this returns the 'excess_on_hand' Quantity planned as of that Date.

If a Date_Range is passed, this returns the smallest 'excess_on_hand' Quantity planned during that Date_Range.

If a Date_Range and a Logical is passed, this returns the smallest 'excess_on_hand' Quantity planned during that Date_Range, if the Logical is True. If the Logical is False, this returns the largest 'excess_on_hand' Quantity planned during that Date_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

`low_on_hand_profile, low_on_hand_profile (Date_Range) ... a`

List(Profile_Quantity) field of model FIXED_QUANTITY

A profile or a list of changes to the low_on_hand Quantity during the specified Date_Range. Mathematically, low_on_hand is computed using following formula, (Buffer_Plan.safety_target(d) - Buffer_Plan.on_hand(d)).

Properties: command=True Export-Only Field

`low_on_hand (Date), low_on_hand (Date_Range, Logical) ... a` Quantity field of model FIXED_QUANTITY

It is the 'on_hand' Quantity planned below the minimum 'safety_target' Quantity. Mathematically, low_on_hand is computed using following formula, (Buffer_Plan.safety_target - Buffer_Plan.on_hand).

If a Date is passed, this returns the low_on_hand Quantity planned as of that Date.

If a Date_Range is passed, this returns the smallest low_on_hand Quantity planned during that Date_Range.

If a Date_Range and a Logical is passed, this returns the smallest low_on_hand Quantity planned during that Date_Range, if the Logical is True. If the Logical is False, this returns the largest low_on_hand Quantity planned during that Date_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

MULTIPLE -- a flow_policy extension of model Buffer_Plan

Extensions	MULTIPLE Extension
------------	--------------------

A MULTIPLE Buffer maintains a 'min_time' safety time and a 'min_on_hand' safety quantity. See the corresponding 'flow_policy' of the Buffer model for more details.

`max_target_profile, max_target_profile (Date_Range) ... a`

List(Profile_Quantity) field of model MULTIPLE

A profile or a list of changes to max allowable Quantity's during the specified finite Date_Range. Up to 'max_target' Quantity is allowed in this buffer. If the Buffer_Plan.on_hand Quantity goes above Buffer_Plan.max_target Quantity, RHYTHM will detect EXCESS_ON_HAND problem.

Properties: command=True Export-Only Field

`max_target (Date), max_target (Date_Range, Logical) ... a` Quantity field of model MULTIPLE

Maximum target Quantity allowed in this BASIC buffer. If the Buffer_Plan.on_hand Quantity goes above Buffer_Plan.max_target Quantity, RHYTHM will detect EXCESS_ON_HAND problem.

If a Date is passed, this returns the max_target Quantity allowed as of that Date in this buffer.

If a Date_Range is passed, this returns the smallest 'max_target' Quantity allowed during that Date_Range in this buffer.

If a Date_Range and a Logical is passed, this returns the smallest 'max_target' Quantity allowed during that Date_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest 'max_target' Quantity allowed during that Date_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

`safety_target_profile, safety_target_profile (Date_Range) ... a`

List(Profile_Quantity) field of model MULTIPLE

A profile or a list of changes to the 'safety_target' Quantity during the specified Date_Range. This 'safety_target' Quantity is the max of Buffer.min_on_hand and the Quantity required to cover the next 'min_time' of consumption in this Buffer.

For example, consider a Buffer with min_on_hand is set to "100", 'min_time' is set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the safety_target_profile will have three entries, first entry with quantity 100 until 4/1, second entry with quantity 200 starting on 4/1 and third entry with quantity 100 starting on 4/15. In the second period "200" will be maintained in order to cover the next "2 weeks" of out flows.

Extensions**MULTIPLE Extension**

Properties: command=True Export-Only Field

safety_target (Date), safety_target (Date_Range, Logical) -- *a Quantity field of model MULTIPLE*

Safety target is a minimum Quantity this BASIC Buffer is required to maintain based on existing plan. Mathematically, 'safety_target' Quantity is the max of Buffer.min_on_hand and the Quantity required to cover the next 'Buffer.min_time' of consumption in this Buffer.

For example, consider a Buffer with 'min_on_hand' set to "100" and 'min_time' set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the 'safety_target' Quantity on date 4/10 will be 200.

If a Date is passed, this returns the 'safety_target' Quantity required to be maintained as of that Date considering the existing plan.

If a Date_Range is passed, this returns the smallest 'safety_target' Quantity required to be maintained during that Date_Range considering the existing plan.

If a Date_Range and a Logical is passed, this returns the smallest 'safety_target' Quantity required to be maintained during that Date_Range considering the existing plan if the Logical is True. If the Logical is False, this returns the largest 'safety_target' Quantity required to be maintained during that Date_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

cycle_on_hand_profile, cycle_on_hand_profile (Date_Range) -- *a*

List(Profile_Quantity) field of model MULTIPLE

A profile or a list of changes to 'cycle_on_hand' (max(0,0), Buffer_Plan.on_hand - Buffer_Plan.safety_target) Quantity's planned during the specified finite Date_Range in this buffer.

Properties: command=True Export-Only Field

cycle_on_hand (Date), cycle_on_hand (Date_Range, Logical) -- *a Quantity field of model MULTIPLE*

Cycle 'on_hand' (max(0,0), Buffer_Plan.on_hand - Buffer_Plan.safety_target) Quantity planned in this buffer.

If a Date is passed, this returns the 'cycle_on_hand' Quantity planned as of that Date in this buffer.

Extensions**MULTIPLE Extension**

If a Date_Range is passed, this returns the smallest 'cycle_on_hand' Quantity planned during that Date_Range in this buffer.

If a Date_Range and a Logical is passed, this returns the smallest 'cycle_on_hand' Quantity planned during that Date_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest 'max_target' Quantity planned during that Date_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

excess_on_hand_profile, excess_on_hand_profile (Date_Range) -- *a*

List(Profile_Quantity) field of model MULTIPLE

A profile or a list of changes to 'excess_on_hand' Quantity during the specified finite Date_Range. This is the 'Buffer_Plan.on_hand' Quantity above allowable 'Buffer_Plan.max_target' Quantity. Mathematically, 'excess_on_hand' is computed using formula, max(0,0), Buffer_Plan.on_hand - Buffer_Plan.max_target.

Properties: command=True Export-Only Field

excess_on_hand (Date), excess_on_hand (Date_Range, Logical) -- *a Quantity field of model MULTIPLE*

'Excess_on_hand' Quantity planned above the allowable 'Buffer.excess_on_hand' will be renamed to 'Buffer.max_cycle_quantity' in this buffer. Mathematically, 'excess_on_hand' is computed using formula, max(0,0), Buffer_Plan.on_hand - Buffer_Plan.max_target.

If a Date is passed, this returns the 'excess_on_hand' Quantity planned as of that Date.

If a Date_Range is passed, this returns the smallest 'excess_on_hand' Quantity planned during that Date_Range.

If a Date_Range and a Logical is passed, this returns the smallest 'excess_on_hand' Quantity planned during that Date_Range, if the Logical is True. If the Logical is False, this returns the largest 'excess_on_hand' Quantity planned during that Date_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

Extensions	MULTIPLE_FILTER_AND_RANK Extension
------------	------------------------------------

`low_on_hand_profile, low_on_hand_profile (Date_Range) -- a`
List(Profile_Quantity)/field of model MULTIPLE
 A profile or a list of changes to the `low_on_hand` Quantity during the specified Date_Range. Mathematically, `low_on_hand` is computed using following formula,
 $(\text{Buffer_Plan.safety_target}(d) - \text{Buffer_Plan.on_hand}(d))$
 Properties: `command=True` Export-Only Field

`low_on_hand (Date), low_on_hand (Date_Range, Logical) -- a` Quantity/field of model MULTIPLE

It is the `on_hand` Quantity planned below the minimum `safety_target` Quantity. Mathematically, `low_on_hand` is computed using following formula,
 $(\text{Buffer_Plan.safety_target} - \text{Buffer_Plan.on_hand})$

If a Date is passed, this returns the `low_on_hand` Quantity planned as of that Date.

If a Date_Range is passed, this returns the smallest `low_on_hand` Quantity planned during that Date_Range.

If a Date_Range and a Logical is passed, this returns the smallest `low_on_hand` Quantity planned during that Date_Range, if the Logical is True. If the Logical is False, this returns the largest `low_on_hand` Quantity planned during that Date_Range.

The Quantity is converted to the unit of this buffer.

Properties: `Export-Only` Field

MULTIPLE_FILTER_AND_RANK -- a flow_policy extension of model Buffer_Plan

A MULTIPLE_FILTER_AND_RANK Buffer maintains a `min_time` safety time and a `min_on_hand` safety quantity. See the corresponding `Flow_policy` of the Buffer model for more details.

`max_target_profile, max_target_profile (Date_Range) -- a`

List(Profile_Quantity)/field of model MULTIPLE_FILTER_AND_RANK
 A profile or a list of changes to max allowable Quantity during the specified finite Date_Range. Up to `max_target` Quantity is allowed in this buffer. If the `Buffer_Plan.on_hand` Quantity goes above `Buffer_Plan.max_target` Quantity, RHYTHM will detect EXCESS_ON_HAND problem.
 Properties: `command=True` Export-Only Field

Extensions	MULTIPLE_FILTER_AND_RANK Extension
------------	------------------------------------

`max_target (Date), max_target (Date_Range, Logical) -- a` Quantity/field of model MULTIPLE_FILTER_AND_RANK
 Maximum target Quantity allowed in this BASIC buffer. If the `Buffer_Plan.on_hand` Quantity goes above `Buffer_Plan.max_target` Quantity, RHYTHM will detect EXCESS_ON_HAND problem.

If a Date is passed, this returns the `max_target` Quantity allowed as of that Date in this buffer.

If a Date_Range is passed, this returns the smallest `max_target` Quantity allowed during that Date_Range in this buffer.

If a Date_Range and a Logical is passed, this returns the smallest `max_target` Quantity allowed during that Date_Range if the Logical is True. If the Logical is False, this returns the largest `max_target` Quantity allowed during that Date_Range.

The Quantity is converted to the unit of this buffer.

Properties: `Export-Only` Field

`safety_target_profile, safety_target_profile (Date_Range) -- a`

List(Profile_Quantity)/field of model MULTIPLE_FILTER_AND_RANK
 A profile or a list of changes to the `safety_target` Quantity during the specified Date_Range. This `safety_target` Quantity is the max of `Buffer.min_on_hand` and the Quantity required to cover the next `min_time` of consumption in this Buffer.

For example, consider a Buffer with `min_on_hand` is set to "100", `min_time` is set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the `safety_target_profile` will have three entries, first entry with quantity 100 until 4/1, second entry with quantity 200 starting on 4/1 and third entry with quantity 100 starting on 4/15. In the second period "200" will be maintained in order to cover the next "2 weeks" of out flows.
 Properties: `command=True` Export-Only Field

`safety_target (Date), safety_target (Date_Range, Logical) -- a` Quantity/field of model MULTIPLE_FILTER_AND_RANK

Safety target is a minimum Quantity this BASIC Buffer is required to maintain based on existing plan. Mathematically, `safety_target` Quantity is the max of `Buffer.min_on_hand` and the Quantity required to cover the next `Buffer.min_time` of consumption in this Buffer.

Extensions **MULTIPLE_FILTER_AND_RANK Extension**

For example, consider a Buffer with min_on_hand 'set' to "100" and min_time 'set' to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the safety_target Quantity on date 4/10 will be 200.

If a Date is passed, this returns the safety_target Quantity required to be maintained as of that Date considering the existing plan.

If a Date_Range is passed, this returns the smallest safety_target Quantity required to be maintained during that Date_Range considering the existing plan.

If a Date_Range and a Logical is passed, this returns the smallest safety_target Quantity required to be maintained during that Date_Range considering the existing plan if the Logical is True. If the Logical is False, this returns the largest safety_target Quantity required to be maintained during that Date_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

cycle_on_hand_profile, cycle_on_hand (Date_Range) -- a

List(Profile_Quantity) field of model MULTIPLE_FILTER_AND_RANK

A profile or a list of changes to cycle_on_hand' (max(0,0, Buffer_Plan_on_hand - Buffer_Plan_safety_target)) Quantity's planned during the specified finite Date_Range in this buffer.

Properties: command=True Export-Only Field

cycle_on_hand (Date), cycle_on_hand (Date_Range Logical) -- a Quantity

field of model MULTIPLE_FILTER_AND_RANK

Cycle_on_hand' (max(0,0, Buffer_Plan_on_hand - Buffer_Plan_safety_target)) Quantity planned in this buffer.

If a Date is passed, this returns the 'cycle_on_hand' Quantity planned as of that Date in this buffer.

If a Date_Range is passed, this returns the smallest 'cycle_on_hand' Quantity planned during that Date_Range in this buffer.

If a Date_Range and a Logical is passed, this returns the smallest 'cycle_on_hand' Quantity planned during that Date_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest 'max_target' Quantity planned during that Date_Range.

Extensions **MULTIPLE_FILTER_AND_RANK Extension**

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

excess_on_hand_profile, excess_on_hand_profile (Date_Range) -- a

List(Profile_Quantity) field of model MULTIPLE_FILTER_AND_RANK

A profile or a list of changes to excess_on_hand Quantity during the specified finite Date_Range. This is the Buffer_Plan_on_hand Quantity above allowable

Buffer_Plan_max_target Quantity. Mathematically, 'excess_on_hand' is computed using formula, max(0,0, Buffer_Plan_on_hand - Buffer_Plan_max_target).

Properties: command=True Export-Only Field

excess_on_hand (Date), excess_on_hand (Date_Range Logical) -- a Quantity

field of model MULTIPLE_FILTER_AND_RANK

Excess_on_hand Quantity planned above the allowable Buffer_excess_on_hand (will be renamed to Buffer_max_cycle_quantity) Quantity in this buffer. Mathematically, 'excess_on_hand' is computed using formula, max(0,0, Buffer_Plan_on_hand - Buffer_Plan_max_target).

If a Date is passed, this returns the 'excess_on_hand' Quantity planned as of that Date.

If a Date_Range is passed, this returns the smallest 'excess_on_hand' Quantity planned during that Date_Range.

If a Date_Range and a Logical is passed, this returns the smallest 'excess_on_hand' Quantity planned during that Date_Range, if the Logical is True. If the Logical is False, this returns the largest 'excess_on_hand' Quantity planned during that Date_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

low_on_hand_profile, low_on_hand_profile (Date_Range) -- a

List(Profile_Quantity) field of model MULTIPLE_FILTER_AND_RANK

A profile or a list of changes to the 'low_on_hand' Quantity during the specified Date_Range. Mathematically, 'low_on_hand' is computed using following formula, (Buffer_Plan_safety_target(d) - Buffer_Plan_on_hand(d)).

Properties: command=True Export-Only Field

Extensions	FIXED_QUANTITY_FENCED Extension
------------	---------------------------------

low_on_hand (Date), low_on_hand (Date, Range, Logical) -- *a* Quantity field of model MULTIPLE_FILTER_AND_RANK

It is the 'on_hand' Quantity planned below the minimum 'safety_target' Quantity. Mathematically, 'low_on_hand' is computed using following formula,
(Buffer_Plan.safety_target - Buffer_Plan.on_hand).

If a Date is passed, this returns the 'low_on_hand' Quantity planned as of that Date.

If a Date_Range is passed, this returns the smallest 'low_on_hand' Quantity planned during that Date_Range.

If a Date_Range and a Logical is passed, this returns the smallest 'low_on_hand' Quantity planned during that Date_Range, if the Logical is True. If the Logical is False, this returns the largest 'low_on_hand' Quantity planned during that Date_Range.

The Quantity is converted to the unit of this buffer:

Properties: Export-Only Field

FIXED_QUANTITY_FENCED -- *a* flow_policy extension of model Buffer_Plan

A FIXED_QUANTITY_FENCED Buffer maintains a 'min_time' safety time and a 'min_on_hand' safety quantity. See the corresponding 'flow_policy' of the Buffer model for more details.

max_target_profile, max_target_profile (Date, Range) -- *a*

List(Profile_Quantity) field of model FIXED_QUANTITY_FENCED

A profile or a list of changes to max allowable Quantity's during the specified finite Date_Range. Up to 'max_target' Quantity is allowed in this buffer. If the Buffer_Plan.on_hand Quantity goes above Buffer_Plan.max_target Quantity, RHYTHM will detect EXCESS_ON_HAND problem.

Properties: command=True Export-Only Field

max_target (Date), max_target (Date, Range, Logical) -- *a* Quantity field of model FIXED_QUANTITY_FENCED

Maximum target Quantity allowed in this BASIC buffer. If the Buffer_Plan.on_hand Quantity goes above Buffer_Plan.max_target Quantity, RHYTHM will detect EXCESS_ON_HAND problem.

If a Date is passed, this returns the 'max_target' Quantity allowed as of that Date in this buffer.

Extensions	FIXED_QUANTITY_FENCED Extension
------------	---------------------------------

If a Date_Range is passed, this returns the smallest 'max_target' Quantity allowed during that Date_Range in this buffer.

If a Date_Range and a Logical is passed, this returns the smallest 'max_target' Quantity allowed during that Date_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest 'max_target' Quantity allowed during that Date_Range.

The Quantity is converted to the unit of this buffer:

Properties: Export-Only Field

safety_target_profile, safety_target_profile (Date, Range) -- *a*

List(Profile_Quantity) field of model FIXED_QUANTITY_FENCED

A profile or a list of changes to the 'safety_target' Quantity during the specified Date_Range. This 'safety_target' Quantity is the max of Buffer.min_on_hand and the Quantity required to cover the next 'min_time' of consumption in this Buffer.

For example, consider a Buffer with 'min_on_hand' set to "100", 'min_time' is set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the 'safety_target_profile' will have three entries: first entry with quantity 100 until 4/1, second entry with quantity 200 starting on 4/1 and third entry with quantity 100 starting on 4/15. In the second period "200" will be maintained in order to cover the next "2 weeks" of out flows.

Properties: command=True Export-Only Field

safety_target (Date), safety_target (Date, Range, Logical) -- *a* Quantity field of model FIXED_QUANTITY_FENCED

Safety target is a minimum Quantity this BASIC Buffer is required to maintain based on existing plan. Mathematically, 'safety_target' Quantity is the max of Buffer.min_on_hand and the Quantity required to cover the next 'Buffer.min_time' of consumption in this Buffer.

For example, consider a Buffer with 'min_on_hand' set to "100" and 'min_time' set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the 'safety_target' Quantity on date 4/10 will be 200.

If a Date is passed, this returns the 'safety_target' Quantity required to be maintained as of that Date considering the existing plan.

If a Date_Range is passed, this returns the smallest 'safety_target' Quantity required to be maintained during that Date_Range considering the existing plan.

If a **Date_Range** and a **Logical** is passed, this returns the smallest 'safety_target' Quantity required to be maintained during that **Date_Range** considering the existing plan if the **Logical** is **True**. If the **Logical** is **False**, this returns the largest 'safety_target' Quantity required to be maintained during that **Date_Range**.

The Quantity is converted to the unit of this buffer:

Properties: Export-Only Field

cycle_on_hand_profile, **cycle_on_hand_profile (Date_Range)** ... *a*

List(Profile_Quantity) field of model FIXED_QUANTITY_FENCED

A profile or a list of changes to **cycle_on_hand** (max(0.0, Buffer_Plan_on_hand - Buffer_Plan_safety_target)) Quantity's planned during the specified finite **Date_Range** in this buffer.

Properties: command=True Export-Only Field

cycle_on_hand (Date), **cycle_on_hand (Date_Range, Logical)** ... *a* Quantity **field of model FIXED_QUANTITY_FENCED**

Cycle_on_hand (max(0.0, Buffer_Plan_on_hand - Buffer_Plan_safety_target)) Quantity planned in this buffer.

If a **Date** is passed, this returns the **cycle_on_hand** Quantity planned as of that **Date** in this buffer.

If a **Date_Range** is passed, this returns the smallest **cycle_on_hand** Quantity planned during that **Date_Range** in this buffer.

If a **Date_Range** and a **Logical** is passed, this returns the smallest 'cycle_on_hand' Quantity planned during that **Date_Range** in this buffer if the **Logical** is **True**. If the **Logical** is **False**, this returns the largest 'max_target' Quantity planned during that **Date_Range**.

The Quantity is converted to the unit of this buffer:

Properties: Export-Only Field

excess_on_hand_profile, **excess_on_hand_profile (Date_Range)** ... *a*

List(Profile_Quantity) field of model FIXED_QUANTITY_FENCED

A profile or a list of changes to **excess_on_hand** Quantity during the specified finite **Date_Range**. This is the **Buffer_Plan_on_hand** Quantity above allowable **Buffer_Plan_max_target** Quantity. Mathematically, **excess_on_hand** is computed using formula: max(0.0, Buffer_Plan_on_hand - Buffer_Plan_max_target).

Properties: command=True Export-Only Field

excess_on_hand (Date), **excess_on_hand (Date_Range, Logical)** ... *a* Quantity **field of model FIXED_QUANTITY_FENCED**

Excess_on_hand Quantity planned above the allowable **Buffer_excess_on_hand** (will be renamed to **Buffer_max_cycle_quantity**) Quantity in this buffer. Mathematically, **excess_on_hand** is computed using formula: max(0.0, Buffer_Plan_on_hand - Buffer_Plan_max_target).

If a **Date** is passed, this returns the **excess_on_hand** Quantity planned as of that **Date**.

If a **Date_Range** is passed, this returns the smallest **excess_on_hand** Quantity planned during that **Date_Range**.

If a **Date_Range** and a **Logical** is passed, this returns the smallest 'excess_on_hand' Quantity planned during that **Date_Range**. If the **Logical** is **True**, if the **Logical** is **False**, this returns the largest 'excess_on_hand' Quantity planned during that **Date_Range**.

The Quantity is converted to the unit of this buffer:

Properties: Export-Only Field

low_on_hand_profile, **low_on_hand_profile (Date_Range)** ... *a*

List(Profile_Quantity) field of model FIXED_QUANTITY_FENCED

A profile or a list of changes to the **low_on_hand** Quantity during the specified **Date_Range**. Mathematically, **low_on_hand** is computed using following formula: (Buffer_Plan_safety_target(d) - Buffer_Plan_on_hand(d)).

Properties: command=True Export-Only Field

low_on_hand (Date), **low_on_hand (Date_Range, Logical)** ... *a* Quantity **field of model FIXED_QUANTITY_FENCED**

It is the **on_hand** Quantity planned below the minimum 'safety_target' Quantity. Mathematically, **low_on_hand** is computed using following formula: (Buffer_Plan_safety_target - Buffer_Plan_on_hand).

If a **Date** is passed, this returns the **low_on_hand** Quantity planned as of that **Date**.

If a **Date_Range** is passed, this returns the smallest **low_on_hand** Quantity planned during that **Date_Range**.

If a Date_Range and a Logical is passed, this returns the smallest 'low_on_hand' Quantity planned during that Date_Range, if the Logical is True. If the Logical is False, this returns the largest 'low_on_hand' Quantity planned during that Date_Range.

The Quantity is converted to the unit of this buffer:

Properties: Export-Only Field

FIXED_TIME -- a flow_policy extension of model Buffer_Plan

A FIXED_TIME Buffer maintains a 'min_time' safety time and a 'min_on_hand' safety quantity. See the corresponding 'flow_policy' of the Buffer model for more details.

max_target_profile, max_target_profile(Date_Range) -- a

List(Profile_Quantity) field of model FIXED_TIME

A profile or a list of changes to max allowable Quantity's during the specified finite Date_Range. Up to 'max_target' Quantity is allowed in this buffer. If the Buffer_Plan.on_hand' Quantity goes above Buffer_Plan.max_target' Quantity, RHYTHM will detect EXCESS_ON_HAND problem.

Properties: command=True Export-Only Field

max_target(Date), max_target(Date_Range, Logical) -- a Quantity field of model FIXED_TIME

Maximum target Quantity allowed in this BASIC buffer. If the Buffer_Plan.on_hand' Quantity goes above Buffer_Plan.max_target' Quantity, RHYTHM will detect EXCESS_ON_HAND problem.

If a Date is passed, this returns the 'max_target' Quantity allowed as of that Date in this buffer.

If a Date_Range is passed, this returns the smallest 'max_target' Quantity allowed during that Date_Range in this buffer.

If a Date_Range and a Logical is passed, this returns the smallest 'max_target' Quantity allowed during that Date_Range in this buffer. If the Logical is True, if the Logical is False, this returns the largest 'max_target' Quantity allowed during that Date_Range.

The Quantity is converted to the unit of this buffer:

Properties: Export-Only Field

safety_target_profile, safety_target_profile(Date_Range) -- a

List(Profile_Quantity) field of model FIXED_TIME

A profile or a list of changes to the 'safety_target' Quantity during the specified Date_Range. This 'safety_target' Quantity is the max of Buffermin_on_hand and the Quantity required to cover the next 'min_time' of consumption in this Buffer.

For example, consider a Buffer with 'min_on_hand' set to "100", 'min_time' is set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the 'safety_target_profile' will have three entries, first entry with quantity 100 until 4/1, second entry with quantity 200 starting on 4/1 and third entry with quantity 100 starting on 4/15. In the second period "200" will be maintained in order to cover the next "2 weeks" of out flows.

Properties: command=True Export-Only Field

safety_target(Date), safety_target(Date_Range, Logical) -- a Quantity field of model FIXED_TIME

Safety target is a minimum Quantity this BASIC Buffer is required to maintain based on existing plan. Mathematically, 'safety_target' Quantity is the max of Buffermin_on_hand and the Quantity required to cover the next 'Buffermin_time' of consumption in this Buffer.

For example, consider a Buffer with 'min_on_hand' set to "100" and 'min_time' set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the 'safety_target' Quantity on date 4/10 will be 200.

If a Date is passed, this returns the 'safety_target' Quantity required to be maintained as of that Date considering the existing plan.

If a Date_Range is passed, this returns the smallest 'safety_target' Quantity required to be maintained during that Date_Range considering the existing plan.

If a Date_Range and a Logical is passed, this returns the smallest 'safety_target' Quantity required to be maintained during that Date_Range considering the existing plan if the Logical is True. If the Logical is False, this returns the largest 'safety_target' Quantity required to be maintained during that Date_Range.

The Quantity is converted to the unit of this buffer:

Properties: Export-Only Field

Extensions	FIXED_TIME Extension
------------	----------------------

cycle_on_hand_profile, cycle_on_hand_profile (Date_Range) -- a

List(Profile_Quantity) field of model FIXED_TIME

A profile or a list of changes to cycle_on_hand' (max(0.0, Buffer_Plan.on_hand - Buffer_Plan.safety_target)) Quantity's planned during the specified finite Date_Range in this buffer.

Properties: command=True Export-Only Field

cycle_on_hand (Date), cycle_on_hand (Date_Range, Logical) -- a Quantity field of model FIXED_TIME

Cycle_on_hand' (max(0.0, Buffer_Plan.on_hand - Buffer_Plan.safety_target)) Quantity planned in this buffer.

If a Date is passed, this returns the cycle_on_hand' Quantity planned as of that Date in this buffer.

If a Date_Range is passed, this returns the smallest cycle_on_hand' Quantity planned during that Date_Range in this buffer.

If a Date_Range and a Logical is passed, this returns the smallest cycle_on_hand' Quantity planned during that Date_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest max_target' Quantity planned during that Date_Range.

The Quantity is converted to the unit' of this buffer'.

Properties: Export-Only Field

excess_on_hand_profile, excess_on_hand_profile (Date_Range) -- a

List(Profile_Quantity) field of model FIXED_TIME

A profile or a list of changes to excess_on_hand' Quantity during the specified finite Date_Range. This is the Buffer_Plan.on_hand' Quantity above allowable Buffer_Plan.max_target Quantity. Mathematically, excess_on_hand' is computed using formula, max(0.0, Buffer_Plan.on_hand - Buffer_Plan.max_target).

Properties: command=True Export-Only Field

excess_on_hand (Date), excess_on_hand (Date_Range, Logical) -- a Quantity field of model FIXED_TIME

Excess_on_hand' Quantity planned above the allowable Buffer.excess_on_hand' will be renamed to Buffer.max_cycle_quantity' Quantity in this buffer. Mathematically, excess_on_hand' is computed using formula, max(0.0, Buffer_Plan.on_hand - Buffer_Plan.max_target).

Extensions	FIXED_TIME Extension
------------	----------------------

If a Date is passed, this returns the excess_on_hand' Quantity planned as of that Date.

If a Date_Range is passed, this returns the smallest excess_on_hand' Quantity planned during that Date_Range.

If a Date_Range and a Logical is passed, this returns the smallest excess_on_hand' Quantity planned during that Date_Range, if the Logical is True. If the Logical is False, this returns the largest excess_on_hand' Quantity planned during that Date_Range.

The Quantity is converted to the unit' of this buffer'.

Properties: Export-Only Field

low_on_hand_profile, low_on_hand_profile (Date_Range) -- a

List(Profile_Quantity) field of model FIXED_TIME

A profile or a list of changes to the low_on_hand' Quantity during the specified Date_Range. Mathematically, low_on_hand' is computed using following formula, (Buffer_Plan.safety_target(d) - Buffer_Plan.on_hand(d)).

Properties: command=True Export-Only Field

low_on_hand (Date), low_on_hand (Date_Range, Logical) -- a Quantity field of model FIXED_TIME

It is the low_on_hand' Quantity planned below the minimum safety_target' Quantity. Mathematically, low_on_hand' is computed using following formula, (Buffer_Plan.safety_target - Buffer_Plan.on_hand).

If a Date is passed, this returns the low_on_hand' Quantity planned as of that Date.

If a Date_Range is passed, this returns the smallest low_on_hand' Quantity planned during that Date_Range.

If a Date_Range and a Logical is passed, this returns the smallest low_on_hand' Quantity planned during that Date_Range, if the Logical is True. If the Logical is False, this returns the largest low_on_hand' Quantity planned during that Date_Range.

The Quantity is converted to the unit' of this buffer'.

Properties: Export-Only Field

<i>Extensions</i>	<i>Forecast Extensions</i>
-------------------	----------------------------

10.19 Forecast Extensions

10.19.1 level extensions of model Forecast

INDIVIDUAL -- a level extension of model Forecast

An INDIVIDUAL Forecast is for an individual Product. The 'product' defines how this Forecast is converted into Requests upon this Site.

The INDIVIDUAL model has these submodels :
ATP_Entry.

The INDIVIDUAL model has fields that references these models :

Product, ATP_Entry.

product -- a Product field of model INDIVIDUAL

The Product for which demand this forecasts.

Properties: standard=True Export-Only Field

atp_entries -- a list of ATP_Entry submodels of model INDIVIDUAL

In each Forecast, there is one ATP_Entry for each Date_Range in the Seller_Plan's list of Date_Ranges. 'atp_horizon'. Each ATP_Entry has the Date_Range for which it maintains allocations, an 'allocated' value that it can promise, a 'consumed' value that maintains the amount already promised, and an 'atp' value that is still available to promise.

GROUP -- a level extension of model Forecast

A GROUP Forecast is for a Product_Group, an aggregation of several Products' Forecasts. The 'product_group' defines how this Forecast is propagated down to the Product_Group's 'sub_groups' and 'products'.

The GROUP model has fields that references these models :

Product_Group.

product_group -- a Product_Group field of model GROUP

The Product for which demand this forecasts.

Properties: standard=True Export-Only Field

sub_forecasts -- a List(Forecast) field of model GROUP

The Forecasts for the 'sub_groups' and 'products' of this Forecast's 'product_group'.
Properties: standard=True Export-Only Field

<i>Extensions</i>	<i>GROUP Extension</i>
-------------------	------------------------

active_sub_forecasts -- a List(Forecast) field of model GROUP

The Forecasts for the 'sub_groups' and 'active_products' of this Forecast's

'product_group'.

Properties: standard=True Export-Only Field

active_leaf_product_forecasts -- a List(Forecast) field of model GROUP

The Forecasts for the 'active_products' of this Forecast's 'product_group'. Returns all active descendants of this GROUP Forecast that are INDIVIDUAL forecasts. In other terms, these are the active 'leaf' Forecasts of this product sub-hierarchy.

Properties: standard=True Export-Only Field

use_std_split -- a Logical field of model GROUP

This field is obsolete. You must now set the 'use_std_split' field in the Product_Group. Note that setting the field at the Product_Group is not exactly equivalent. The split

now applies to all Forecasts of the Product_Group.

Properties: obsolete=True Export-Only Field

10.20 Forecast_Entry Extensions

10.20.1 forecast_policy extensions of model Forecast_Entry

10.20.2 allocation_policy extensions of model Forecast_Entry

MEMBER_RANK -- a allocation_policy extension of model Forecast_Entry

The MEMBER_RANK allocation_policy distributes the allocations to the 'members' in the order of their 'rank'. For Sellers with equal 'rank', it distributes to each proportional to the quantity committed to by that Seller. A portion of the allocation can be retained for use at the discretion of this Seller. Any leftovers due to lot sizing or explicit adjustments will be available on a first-come-first-served (FCFS) basis.

pooled_allocated -- a Quantity field of model MEMBER_RANK
collective allocated quantity for members whose 'lock_allocated' field is set to true
Default: 0

pooled_accepted -- a Quantity field of model MEMBER_RANK
collective accepted quantity for members whose 'lock_allocated' field is set to true
Default: 0

pooled_allocated_available -- a Quantity field of model MEMBER_RANK
collective allocated_available quantity for members whose 'lock_allocated' field is set to true
Properties: Export-Only Field

pooled_available -- a Quantity field of model MEMBER_RANK
collective accepted_available quantity for members whose 'lock_allocated' field is set to true
Properties: Export-Only Field

10.21 Site_Plan Extensions

10.21.1 role extensions of model Site_Plan

LINK -- a role extension of model Site_Plan

A LINK Site is considered to be a "link" of this supply "chain".

The LINK model has these submodels:
Buffer_Plan, Resource_Plan, Operation_Plan, Operation_State, Request, Promise, Acceptance.

The LINK model has fields that references these models:
Buffer_Plan, Resource_Plan, Operation_Plan, Operation_State, Request, Promise, Acceptance.

buffer_plans -- a list of Buffer_Plan submodels of model LINK
The plans for each Buffer defined in the site_plan site. Each buffer in each Site will have exactly one Buffer_Plan per Site_Plan.

resource_plans -- a list of Resource_Plan submodels of model LINK
The plans for each Resource defined in this site_plan's site. Each resource in each Site will have one Resource_Plan per Site_Plan.

operation_plans -- a list of Operation_Plan submodels of model LINK
The plans for each Operation defined in this site_plan's site. Each operation in each Site could have multiple Operation_Plans per Site_Plan.

operation_states -- a list of Operation_State submodels of model LINK
State reports which will be assigned to Operation_Plans.

requests -- a list of Request submodels of model LINK
The Requests that have been made by other Sites for items supplied by this Site. This is the raw demand on this Site.

promises -- a list of Promise submodels of model LINK
The promises that have been made by this Site to supply items to other Sites. This is the demand this Site has agreed to fulfill. By some definitions, this is the master production schedule.

Extensions**LINK Extension****acceptances - - a list of Acceptance submodels of model LINK**

The acceptances that have been made by the other Site in response to the promises from this site

plan_to_satisfy_unanswered_requests - - a Void field of model LINK

This command creates plans to satisfy all the Requests that have been made upon this Site_Plan that have not yet been "answered". It does this by simply calling plan_to_satisfy on each of its "unanswered" requests.

Alternatively, plan_to_satisfy_all_promises can create plans such that all Promises would be satisfied; or plan_to_satisfy_all_requests to create plans such that all Requests upon this Site_Plan are satisfied.

Properties: command=True Export-Only Field

plan_to_satisfy_queued_requests - - a Void field of model LINK

This command creates plans to satisfy all the Requests that have been made upon this Site_Plan that have been "queued" for later consideration. It does this by simply calling plan_to_satisfy on each of its "queued" requests.

Alternatively, plan_to_satisfy_all_promises can create plans such that all Promises would be satisfied; or plan_to_satisfy_all_requests to create plans such that all Requests upon this Site_Plan are satisfied.

Properties: command=True Export-Only Field

plan_to_satisfy_all_requests - - a Void field of model LINK

This command creates plans to satisfy all the Requests that have been made upon this Site_Plan. It does this by simply calling plan_to_satisfy on each of its requests.

Thus, it is equivalent to requests_for_each(plan_to_satisfy).

Alternatively, plan_to_satisfy_all_promises can create plans such that all Promises would be satisfied; or plan_to_satisfy_unanswered_requests to create plans such that Requests that have not yet been "answered" are satisfied.

Properties: command=True Export-Only Field

plan_to_satisfy_all_promises - - a Void field of model LINK

This command creates plans to satisfy all the Promises that have been made by this Site_Plan. It does this by simply calling plan_to_satisfy on each of its promises.

Thus, it is equivalent to promises_for_each(plan_to_satisfy).

Extensions**LINK Extension**

Alternatively, plan_to_satisfy_all_requests can create plans such that all Requests would be satisfied; or plan_to_satisfy_unanswered_requests to create plans such that Requests that have not yet been "answered" are satisfied.

Properties: command=True Export-Only Field

promise_as_planned - - a Void field of model LINK

This command sets this Site_Plan's promises to promise what has been planned for each. It does this by simply calling promise_as_planned on each of its promises. Thus, it is equivalent to promises_for_each(promise_as_planned).

Properties: command=True Export-Only Field

supply_requests - - a List(Request) field of model LINK

The Requests that have been made by this Site for Items supplied by other Sites.

Properties: Export-Only Field

supply_promises - - a List(Promise) field of model LINK

The Promises that have been made by other Sites to supply Items to this Site.

Properties: Export-Only Field

supply_item_requests, supply_item_requests(Date_Range),

supply_item_requests(item), supply_item_requests(item, Date_Range) - - a List(Item, Request) field of model LINK

The Requests that have been made by this Site for Items supplied by other Sites.

Properties: Export-Only Field

item_demand(List(Item), Date_Range) - - a Quantity field of model LINK

Returns the sum of demand for a List of Items during a Date_Range. The Quantity will be unitless as the sum of the number of units of each Item.

Properties: Export-Only Field

problems, problems(Date_Range), problems(Problem_Category), problems(Problem_Category, Date_Range) - - a List(Problem) field of model LINK

The Problems detected with this Site_Plan. If passed a Problem_Category, only the Problems with that category are returned. If passed a Date_Range, only the Problems whose dates overlap are returned.

Note that you can pass in one of the special Problem_Category's to get Problems in larger groups. For example, the OPERATION Problem_Category will give all Problems in Operation-related Problem categories. Similarly for RESOURCE, BUFFER, and even ALL.

Properties: Export-Only Field

problem_categories - - a List(Problem_Category) field of model LINK
For each different category of Problem in problems, a Problem_Category is added to this list. It gives you the name of the category (in various forms) plus a list of just the Problems of that category. These sublists are often much easier to deal with than the full problems list.

Note that this List will not include special Problem_Categories, such as OPERATION, RESOURCE, BUFFER, or ALL.
Properties: Export-Only Field

SUPPLIER -- a role extension of model Site_Plan

A SUPPLIER Site is not planned or modeled in detail; rather it represents the Items that can be procured from a supplier by LINK Sites, the Requests for those Items, and the Promises received from the supplier.

The SUPPLIER model has these submodels :
Request, Promise, Acceptance.

The SUPPLIER model has fields that references these models :
Request, Promise, Acceptance.

requests - - a list of Request submodels of model SUPPLIER
The Requests that have been made by other Sites for Items supplied by this Site. This is the raw demand on this Site.

promises - - a list of Promise submodels of model SUPPLIER
The promises that have been made by this Site to supply Items to other Sites. This is the demand this Site has agreed to fulfill.

acceptances - - a list of Acceptance submodels of model SUPPLIER
The acceptances that have been made by the other Site in response to the promises from this site

CUSTOMER -- a role extension of model Site_Plan

A CUSTOMER Site is not planned or modeled in detail; rather, it is simply a destination for deliveries and source of Requests.

10.22 Problem Extensions

10.22.1 category extensions of model Problem

REQUEST_NOT_PLANNED -- a category extension of model Problem

A REQUEST_NOT_PLANNED Problem indicates that the Item_request was issued after the Item_promise was offered -- and -- the delivery_plan has not been planned to satisfy the Item_request; either there is no delivery_plan or the delivery_plan is for the older Item_promise. This Problem will not occur if the delivery_request has an infinite due Date, the Item_request is for a zero quantity, or the Item_promise was offered after the Item_request was issued.

To resolve this Problem, either eliminate or zero out the Item_request (unlikely), offer a Promise, or plan to satisfy the Item_request which will create and/or replan the delivery_plan to meet the Item_request.

The typical series of events begins with the receipt of a newly issued Request. Its Item_Request will each have this REQUEST_NOT_PLANNED Problem. The planner either uses ATP to Promise them directly (eliminating this problem, replacing it with PROMISE_NOT_PLANNED), or the planner does plan to satisfy (eliminating this Problem, replacing it with PROMISE_NOT_OFFERED).

The REQUEST_QUEUED Problem is a variation of this Problem -- both are resolved the same way, but a REQUEST_QUEUED Request has been considered before, whereas a REQUEST_NOT_PLANNED Request is new.

This is in the REQUEST Problem_Set category, as well as
REQUEST_NOT_PLANNED.

The REQUEST_NOT_PLANNED model has fields that references these models :
Item_Request, Item_Promise, Item_Acceptance, Delivery_Request,
Delivery_Promise, Delivery_Acceptance, Operation_Plan.

Item_request - - a Item_Request field of model REQUEST_NOT_PLANNED
The Item_Request that has this REQUEST_NOT_PLANNED Problem. This Item_request.delivery_plan either is nonexistent or is planned for this Item_request.item_promise, which is older than this Item_Request.
Properties: Export-Only Field

Item_promise - - a Item_Promise field of model REQUEST_NOT_PLANNED
The Item_Promise for the Item_request.

Properties: Export-Only Field

item_acceptance -- a Item_Acceptance field of model REQUEST_NOT_PLANNED
The Item_Acceptance for the item_request.
Properties: Export-Only Field

delivery_request -- a Delivery_Request field of model REQUEST_NOT_PLANNED
This is simply shorthand for item_request.owner.
Properties: Export-Only Field

delivery_promise -- a Delivery_Promise field of model REQUEST_NOT_PLANNED
This is simply shorthand for item_promise.owner.
Properties: Export-Only Field

delivery_acceptance -- a Delivery_Acceptance field of model REQUEST_NOT_PLANNED
This is simply shorthand for item_acceptance.owner.
Properties: Export-Only Field

request -- a Request field of model REQUEST_NOT_PLANNED
This is simply shorthand for item_request.delivery_request.owner.
Properties: Export-Only Field

promise -- a Promise field of model REQUEST_NOT_PLANNED
This is simply shorthand for item_promise.delivery_promise.owner.
Properties: Export-Only Field

acceptance -- a Acceptance field of model REQUEST_NOT_PLANNED
This is simply shorthand for item_acceptance.delivery_acceptance.owner.
Properties: Export-Only Field

delivery_plan -- a Operation_Plan field of model REQUEST_NOT_PLANNED
This is simply shorthand for item_request.delivery_plan.
Properties: Export-Only Field

REQUEST_PLANNED_LATE -- a category extension of model Problem

A REQUEST_PLANNED_LATE Problem indicates that the delivery_plan has been planned to satisfy the item_request but is in fact planned later than the delivery_request's due Dates. This Problem will not occur if the delivery_request has an infinite due Date.

To resolve this Problem, either set the delivery_request.due to be later (unlikely), or adjust the delivery_plan such that its end Date is within due.

This is in both the REQUEST and REQUEST_PLAN Problem. Set categories, as well as REQUEST_PLANNED_LATE.

The REQUEST_PLANNED_LATE model has fields that references these models: Item_Request, Item_Promise, Item_Acceptance, Delivery_Request, Delivery_Promise, Delivery_Acceptance, Operation_Plan.

item_request -- a Item_Request field of model REQUEST_PLANNED_LATE
The Item_Request that has this REQUEST_PLANNED_LATE Problem. This item_request.delivery_plan is planned to end later than this item_request.owner.due (the due Date of this Item_Request).
Properties: Export-Only Field

item_promise -- a Item_Promise field of model REQUEST_PLANNED_LATE
The Item_Promise for the item_request.
Properties: Export-Only Field

item_acceptance -- a Item_Acceptance field of model REQUEST_PLANNED_LATE
The Item_Acceptance for the item_request.
Properties: Export-Only Field

delivery_request -- a Delivery_Request field of model REQUEST_PLANNED_LATE
This is simply shorthand for item_request.owner. The delivery_plan is planned to end later than this delivery_request.due.
Properties: Export-Only Field

delivery_promise -- a Delivery_Promise field of model REQUEST_PLANNED_LATE
This is simply shorthand for item_promise.owner.
Properties: Export-Only Field

Extensions	REQUEST_PLANNED_EARLY Extension
------------	---------------------------------

delivery_acceptance - - - a Delivery_Acceptance field of model REQUEST_PLANNED_LATE

This is simply shorthand for 'item_acceptance.owner'.

Properties: Export-Only Field

request - - - a Request field of model REQUEST_PLANNED_LATE

This is simply shorthand for 'item_request.delivery_request.owner'.

Properties: Export-Only Field

promise - - - a Promise field of model REQUEST_PLANNED_LATE

This is simply shorthand for 'item_promise.delivery_promise.owner'.

Properties: Export-Only Field

acceptance - - - a Acceptance field of model REQUEST_PLANNED_LATE

This is simply shorthand for 'item_acceptance.delivery_acceptance.owner'.

Properties: Export-Only Field

delivery_plan - - - a Operation_Plan field of model REQUEST_PLANNED_LATE

This is simply shorthand for 'item_request.delivery_plan'. This 'delivery_plan' is planned to end later than 'delivery_request.due'.

Properties: Export-Only Field

REQUEST_PLANNED_EARLY - - a category extension of model Problem

A REQUEST_PLANNED_EARLY Problem indicates that the 'delivery_plan' has been planned to satisfy the 'delivery_request' but is in fact planned earlier than the 'delivery_request's due' Date. This Problem will not occur if the 'delivery_request' has an infinite 'due' Date.

To resolve this Problem, either set the 'delivery_request.due' to be earlier (unlikely), or adjust the 'delivery_plan' such that its end Date is within 'due'.

This is in both the REQUEST and REQUEST_PLAN Problem. Set categories, as well as REQUEST_PLANNED_EARLY.

The REQUEST_PLANNED_EARLY model has fields that references these models: Item_Request, Item_Promise, Item_Acceptance, Delivery_Request, Delivery_Promise, Delivery_Acceptance, Operation_Plan.

Extensions	REQUEST_PLANNED_EARLY Extension
------------	---------------------------------

item_request - - - a Item_Request field of model REQUEST_PLANNED_EARLY

The Item_Request that has this REQUEST_PLANNED_EARLY Problem. This

'item_request.delivery_plan' is planned to end earlier than this

'item_request.owner.due' (the due Date of this Item_Request).

Properties: Export-Only Field

item_promise - - - a Item_Promise field of model REQUEST_PLANNED_EARLY

The Item_Promise for the 'item_request'.

Properties: Export-Only Field

item_acceptance - - - a Item_Acceptance field of model REQUEST_PLANNED_EARLY

The Item_Acceptance for the 'item_request'.

Properties: Export-Only Field

delivery_request - - - a Delivery_Request field of model REQUEST_PLANNED_EARLY

This is simply shorthand for 'item_request.owner'. The 'delivery_plan' is planned to end earlier than this 'delivery_request.due'.

Properties: Export-Only Field

delivery_promise - - - a Delivery_Promise field of model REQUEST_PLANNED_EARLY

This is simply shorthand for 'item_promise.owner'.

Properties: Export-Only Field

delivery_acceptance - - - a Delivery_Acceptance field of model REQUEST_PLANNED_EARLY

This is simply shorthand for 'item_acceptance.owner'.

Properties: Export-Only Field

request - - - a Request field of model REQUEST_PLANNED_EARLY

This is simply shorthand for 'item_request.delivery_request.owner'.

Properties: Export-Only Field

promise - - - a Promise field of model REQUEST_PLANNED_EARLY

This is simply shorthand for 'item_promise.delivery_promise.owner'.

Properties: Export-Only Field

acceptance - - - a Acceptance field of model REQUEST_PLANNED_EARLY

This is simply shorthand for 'item_acceptance.delivery_acceptance.owner'.

Extensions	REQUEST_PLANNED_SHORT Extension
------------	---------------------------------

Properties: Export-Only Field

delivery_plan - - *a Operation_Plan field of model REQUEST_PLANNED_EARLY*
This is simply shorthand for item_request.delivery_plan. This delivery_plan is planned to end earlier than delivery_request.due.
Properties: Export-Only Field

REQUEST_PLANNED_SHORT - - a category extension of model Problem

A REQUEST_PLANNED_SHORT Problem indicates that the delivery_plan has been planned to satisfy the item_request but is in fact planned for less quantity than requested. This Problem will not occur if the item_request has an infinite quantity range.

To resolve this Problem, either reduce item_request.quantity (unlikely), or increase the Quantity planned by item_request.delivery_plan to be within the requested Quantity_Range.

This is in both the REQUEST and REQUEST_PLAN Problem_Sets categories, as well as REQUEST_PLANNED_SHORT.

The REQUEST_PLANNED_SHORT model has fields that reference these models:
Item_Request, Item_Promise, Item_Acceptance, Delivery_Request,
Delivery_Promise, Delivery_Acceptance, Operation_Plan.

Item_request - - *a Item_Request field of model REQUEST_PLANNED_SHORT*
The Item_Request that has this REQUEST_PLANNED_SHORT Problem. This item_request.quantity range.
Properties: Export-Only Field

Item_promise - - *a Item_Promise field of model REQUEST_PLANNED_SHORT*
The Item_Promise for the item_request.
Properties: Export-Only Field

Item_acceptance - - *a Item_Acceptance field of model REQUEST_PLANNED_SHORT*
The Item_Acceptance for the item_request.
Properties: Export-Only Field

Extensions	REQUEST_PLANNED_EXCESS Extension
------------	----------------------------------

delivery_request - - *a Delivery_Request field of model REQUEST_PLANNED_SHORT*
This is simply shorthand for item_request.owner.
Properties: Export-Only Field

delivery_promise - - *a Delivery_Promise field of model REQUEST_PLANNED_SHORT*
This is simply shorthand for item_promise.owner.
Properties: Export-Only Field

delivery_acceptance - - *a Delivery_Acceptance field of model REQUEST_PLANNED_SHORT*
This is simply shorthand for item_acceptance.owner.
Properties: Export-Only Field

request - - *a Request field of model REQUEST_PLANNED_SHORT*
This is simply shorthand for item_request.delivery_request.owner.
Properties: Export-Only Field

promise - - *a Promise field of model REQUEST_PLANNED_SHORT*
This is simply shorthand for item_promise.delivery_promise.owner.
Properties: Export-Only Field

acceptance - - *a Acceptance field of model REQUEST_PLANNED_SHORT*
This is simply shorthand for item_acceptance.delivery_acceptance.owner.
Properties: Export-Only Field

delivery_plan - - *a Operation_Plan field of model REQUEST_PLANNED_SHORT*
This is simply shorthand for item_request.delivery_plan. This delivery_plan is planned to deliver less than item_request.quantity.
Properties: Export-Only Field

REQUEST_PLANNED_EXCESS - - a category extension of model Problem

A REQUEST_PLANNED_EXCESS Problem indicates that the delivery_plan has been planned to satisfy the item_request but is in fact planned for more quantity than requested. This Problem will not occur if the item_request has an infinite quantity range.

To resolve this Problem, either increase `Item_request.quantity` (unlikely), or reduce the Quantity planned by `Item_request.delivery_plan` to be within the requested `Quantity_Range`.

This is in both the REQUEST and REQUEST_PLAN Problem_Set categories, as well as REQUEST_PLANNED_EXCESS.

The REQUEST_PLANNED_EXCESS model has fields that references these models: `Item_Request`, `Item_Promise`, `Item_Acceptance`, `Delivery_Request`, `Delivery_Promise`, `Delivery_Acceptance`, `Operation_Plan`.

`Item_request` - - - a `Item_Request` field of model REQUEST_PLANNED_EXCESS

The `Item_Request` that has this REQUEST_PLANNED_EXCESS Problem. This `Item_request.delivery_plan` is planned to deliver more quantity than this `Item_request.quantity` range.

Properties: Export-Only Field

`Item_promise` - - - a `Item_Promise` field of model REQUEST_PLANNED_EXCESS
The `Item_Promise` for the `Item_request`.

Properties: Export-Only Field

`Item_acceptance` - - - a `Item_Acceptance` field of model REQUEST_PLANNED_EXCESS

The `Item_Acceptance` for the `Item_request`.

Properties: Export-Only Field

`delivery_request` - - - a `Delivery_Request` field of model REQUEST_PLANNED_EXCESS

This is simply shorthand for `Item_request.owner`.

Properties: Export-Only Field

`delivery_promise` - - - a `Delivery_Promise` field of model REQUEST_PLANNED_EXCESS

This is simply shorthand for `Item_promise.owner`.

Properties: Export-Only Field

`delivery_acceptance` - - - a `Delivery_Acceptance` field of model REQUEST_PLANNED_EXCESS

This is simply shorthand for `Item_acceptance.owner`.

Properties: Export-Only Field

`request` - - - a `Request` field of model REQUEST_PLANNED_EXCESS
This is simply shorthand for `Item_request.delivery_request.owner`.

Properties: Export-Only Field

`promise` - - - a `Promise` field of model REQUEST_PLANNED_EXCESS

This is simply shorthand for `Item_promise.delivery_promise.owner`.

Properties: Export-Only Field

`acceptance` - - - a `Acceptance` field of model REQUEST_PLANNED_EXCESS

This is simply shorthand for `Item_acceptance.delivery_acceptance.owner`.

Properties: Export-Only Field

`delivery_plan` - - - a `Operation_Plan` field of model REQUEST_PLANNED_EXCESS

This is simply shorthand for `Item_request.delivery_plan`. This `delivery_plan` is planned to deliver more than `Item_request.quantity`.

Properties: Export-Only Field

PROMISE_NOT_PLANNED - - - a category extension of model Problem

A PROMISE_NOT_PLANNED Problem indicates that the `delivery_plan` has not been planned to satisfy the `Item_promise`: either there is no `delivery_plan`, or the `delivery_plan` is for the `Item_request` rather than the `Item_promise`. This Problem will not occur if the `delivery_promise` has an infinite 'due' Date or the `Item_promise` is for a zero quantity.

To resolve this Problem, either eliminate or zero out the `Item_promise` (unlikely), or `plan_to_satisfy` the `Item_promise` which will create and/or replan the `delivery_plan` to meet the `Item_promise`, or if the `delivery_plan` is for the `Item_request`, then `promise_as_planned`.

This is in the PROMISE Problem_Set category, as well as PROMISE_NOT_PLANNED.

The PROMISE_NOT_PLANNED model has fields that references these models: `Item_Request`, `Item_Promise`, `Item_Acceptance`, `Delivery_Request`, `Delivery_Promise`, `Delivery_Acceptance`, `Operation_Plan`.

`Item_request` - - - a `Item_Request` field of model PROMISE_NOT_PLANNED

The `Item_Request` that has the PROMISE_NOT_PLANNED problem.

Properties: Export-Only Field

Extensions	PROMISE_NOT_PLANNED Extension
------------	-------------------------------

Item_promise - - - *a Item_Promise field of model PROMISE_NOT_PLANNED*

The Item_Promise that has this PROMISE_NOT_PLANNED Problem. This Item_promise.delivery_plan either is nonexistent or is planned for this Item_promise.item_request.

Properties: Export-Only Field

Item_acceptance - - - *a Item_Acceptance field of model PROMISE_NOT_PLANNED*

The Item_Promise for the Item_request.

Properties: Export-Only Field

delivery_request - - - *a Delivery_Request field of model PROMISE_NOT_PLANNED*

This is simply shorthand for Item_request.owner.

Properties: Export-Only Field

delivery_promise - - - *a Delivery_Promise field of model PROMISE_NOT_PLANNED*

This is simply shorthand for Item_promise.owner.

Properties: Export-Only Field

delivery_acceptance - - - *a Delivery_Acceptance field of model PROMISE_NOT_PLANNED*

This is simply shorthand for Item_acceptance.owner.

Properties: Export-Only Field

request - - - *a Request field of model PROMISE_NOT_PLANNED*

This is simply shorthand for Item_request.delivery_request.owner.

Properties: Export-Only Field

promise - - - *a Promise field of model PROMISE_NOT_PLANNED*

This is simply shorthand for Item_promise.delivery_promise.owner.

Properties: Export-Only Field

acceptance - - - *a Acceptance field of model PROMISE_NOT_PLANNED*

This is simply shorthand for Item_acceptance.delivery_acceptance.owner.

Properties: Export-Only Field

delivery_plan - - - *a Operation_Plan field of model PROMISE_NOT_PLANNED*

This is simply shorthand for Item_promise.delivery_plan.

Properties: Export-Only Field

Extensions	PROMISE_PLANNED_LATE Extension
------------	--------------------------------

PROMISE_PLANNED_LATE - - - *a category extension of model Problem*

A PROMISE_PLANNED_LATE Problem indicates that the 'delivery_plan' has been planned to satisfy the 'Item_promise' but is in fact planned later than the 'delivery_promise's 'due' Dates. This Problem will not occur if the 'delivery_promise' has an infinite 'due' Date.

To resolve this Problem, either set the 'delivery_promise.due' to be later (unlikely), or adjust the 'delivery_plan' such that its end Date is within 'due'.

This is in both the PROMISE and PROMISE_PLAN Problem_Sets categories, as well as PROMISE_PLANNED_LATE.

The PROMISE_PLANNED_LATE model has fields that references these models :
Item_Request, Item_Promise, Item_Acceptance, Delivery_Request,
Delivery_Promise, Delivery_Acceptance, Operation_Plan.

Item_request - - - *a Item_Request field of model PROMISE_PLANNED_LATE*

The Item_Request that has the PROMISE_PLANNED_LATE problem.

Properties: Export-Only Field

Item_promise - - - *a Item_Promise field of model PROMISE_PLANNED_LATE*

The Item_Promise that has this PROMISE_PLANNED_LATE Problem. This Item_promise.delivery_plan is planned to end later than this

Item_promise.owner.due (the due Date of this Item_Promise).

Properties: Export-Only Field

Item_acceptance - - - *a Item_Acceptance field of model PROMISE_PLANNED_LATE*

The Item_Acceptance for the Item_request.

Properties: Export-Only Field

delivery_request - - - *a Delivery_Request field of model PROMISE_PLANNED_LATE*

This is simply shorthand for Item_request.owner.

Properties: Export-Only Field

delivery_promise - - - *a Delivery_Promise field of model PROMISE_PLANNED_LATE*

This is simply shorthand for Item_promise.owner. The 'delivery_plan' is planned to end later than this 'delivery_promise.due'.

Properties: Export-Only Field

delivery_acceptance - - - a Delivery_Acceptance field of model

PROMISE_PLANNED_LATE

This is simply shorthand for 'item_acceptance.owner'.

Properties: Export-Only Field

request - - - a Request field of model **PROMISE_PLANNED_LATE**

This is simply shorthand for 'item_request.delivery_request.owner'.

Properties: Export-Only Field

promise - - - a Promise field of model **PROMISE_PLANNED_LATE**

This is simply shorthand for 'item_promise.delivery_promise.owner'.

Properties: Export-Only Field

acceptance - - - a Acceptance field of model **PROMISE_PLANNED_LATE**

This is simply shorthand for 'item_acceptance.delivery_acceptance.owner'.

Properties: Export-Only Field

delivery_plan - - - a Operation_Plan field of model **PROMISE_PLANNED_LATE**

This is simply shorthand for 'item_promise.delivery_plan'. This *delivery_plan* is planned to end later than *delivery_promise.due*.

Properties: Export-Only Field

PROMISE_PLANNED_EARLY - - - a category extension of model Problem

A **PROMISE_PLANNED_EARLY** Problem indicates that the *delivery_plan* has been planned to satisfy the 'item_promise' but is in fact planned earlier than the *delivery_promise's due* Dates. This Problem will not occur if the *delivery_promise* has an infinite 'due' Date.

To resolve this Problem, either set the *delivery_promise.due* to be earlier (unlikely), or adjust the *delivery_plan* such that its end Date is within *due*.

This is in both the **PROMISE** and **PROMISE_PLAN** Problem Set categories, as well as **PROMISE_PLANNED_EARLY**.

The **PROMISE_PLANNED_EARLY** model has fields that reference these models :
Item_Request, *Item_Promise*, *Item_Acceptance*, *Delivery_Request*,
Delivery_Promise, *Delivery_Acceptance*, *Operation_Plan*.

item_request - - - a Item_Request field of model **PROMISE_PLANNED_EARLY**
The *Item_Request* that has this **PROMISE_PLANNED_EARLY** Problem.
Properties: Export-Only Field

item_promise - - - a Item_Promise field of model **PROMISE_PLANNED_EARLY**
The *Item_Promise* that has this **PROMISE_PLANNED_EARLY** Problem. This *item_promise.delivery_plan* is planned to end earlier than this *item_promise.owner.due* (the due Date of this *item_Promise*).
Properties: Export-Only Field

item_acceptance - - - a Item_Acceptance field of model
PROMISE_PLANNED_EARLY
The *Item_Acceptance* for the *item_request*.
Properties: Export-Only Field

delivery_request - - - a Delivery_Request field of model
PROMISE_PLANNED_EARLY
This is simply shorthand for 'item_request.owner'.

Properties: Export-Only Field

delivery_promise - - - a Delivery_Promise field of model
PROMISE_PLANNED_EARLY
This is simply shorthand for 'item_promise.owner'. The *delivery_plan* is planned to end earlier than this *delivery_promise.due*.

Properties: Export-Only Field

delivery_acceptance - - - a Delivery_Acceptance field of model
PROMISE_PLANNED_EARLY
This is simply shorthand for 'item_acceptance.owner'.

Properties: Export-Only Field

request - - - a Request field of model **PROMISE_PLANNED_EARLY**
This is simply shorthand for 'item_request.delivery_request.owner'.

Properties: Export-Only Field

promise - - - a Promise field of model **PROMISE_PLANNED_EARLY**
This is simply shorthand for 'item_promise.delivery_promise.owner'.

Properties: Export-Only Field

acceptance - - - a Acceptance field of model **PROMISE_PLANNED_EARLY**
This is simply shorthand for 'item_acceptance.delivery_acceptance.owner'.

Properties: Export-Only Field

delivery_plan - - *a Operation_Plan field of model*
PROMISE_PLANNED_EARLY
 This is simply shorthand for 'item_promise.delivery_plan'. This 'delivery_plan' is planned to end earlier than 'delivery_promise.due'.
 Properties: Export-Only Field

PROMISE_PLANNED_SHORT - - *a category extension of model Problem*

A **PROMISE_PLANNED_SHORT** Problem indicates that the 'delivery_plan' has been planned to satisfy the 'item_promise' but is in fact planned for less 'quantity' than promised. This Problem will not occur if the 'item_promise' has an infinite 'quantity' range.

To resolve this Problem, either reduce 'item_promise.quantity' (unlikely), or increase the 'Quantity' planned by 'item_promise.delivery_plan' to be within the promised 'Quantity_Range'.

This is in both the **PROMISE** and **PROMISE_PLAN** Problem_Set categories, as well as **PROMISE_PLANNED_SHORT**.

The **PROMISE_PLANNED_SHORT** model has fields that references these models:

Item_Request, Item_Promise, Item_Acceptance, Delivery_Request,
 Delivery_Promise, Delivery_Acceptance, Operation_Plan.

item_request - - *a Item_Request field of model PROMISE_PLANNED_SHORT*
 The Item_Request that has this **PROMISE_PLANNED_SHORT** Problem.
 Properties: Export-Only Field

item_promise - - *a Item_Promise field of model PROMISE_PLANNED_SHORT*
 The Item_Promise that has this **PROMISE_PLANNED_SHORT** Problem. This 'item_promise.delivery_plan' is planned to deliver less 'quantity' than this 'item_promise.quantity' range.
 Properties: Export-Only Field

item_acceptance - - *a Item_Acceptance field of model*
PROMISE_PLANNED_SHORT
 The Item_Acceptance for the 'item_request'.
 Properties: Export-Only Field

delivery_request - - *a Delivery_Request field of model*
PROMISE_PLANNED_SHORT
 This is simply shorthand for 'item_request.owner'.
 Properties: Export-Only Field

delivery_promise - - *a Delivery_Promise field of model*
PROMISE_PLANNED_SHORT
 This is simply shorthand for 'item_promise.owner'.
 Properties: Export-Only Field

delivery_acceptance - - *a Delivery_Acceptance field of model*
PROMISE_PLANNED_SHORT
 This is simply shorthand for 'item_acceptance.owner'.
 Properties: Export-Only Field

request - - *a Request field of model PROMISE_PLANNED_SHORT*
 This is simply shorthand for 'item_request.delivery_request.owner'.
 Properties: Export-Only Field

promise - - *a Promise field of model PROMISE_PLANNED_SHORT*
 This is simply shorthand for 'item_promise.delivery_promise.owner'.
 Properties: Export-Only Field

acceptance - - *a Acceptance field of model PROMISE_PLANNED_SHORT*
 This is simply shorthand for 'item_acceptance.delivery_acceptance.owner'.
 Properties: Export-Only Field

delivery_plan - - *a Operation_Plan field of model*
PROMISE_PLANNED_SHORT
 This is simply shorthand for 'item_promise.delivery_plan'. This 'delivery_plan' is planned to deliver less than 'item_promise.quantity'.
 Properties: Export-Only Field

PROMISE_PLANNED_EXCESS - - *a category extension of model Problem*

A **PROMISE_PLANNED_EXCESS** Problem indicates that the 'delivery_plan' has been planned to satisfy the 'item_promise' but is in fact planned for more 'quantity' than promised. This Problem will not occur if the 'item_promise' has an infinite 'quantity' range.

<i>Extensions</i>	<i>PROMISE_PLANNED_EXCESS Extension</i>
-------------------	---

To resolve this Problem, either increase 'item_promise.quantity' (unlikely), or reduce the Quantity planned by 'item_promise.delivery_plan' to be within the promised Quantity_Range.

This is in both the PROMISE and PROMISE_PLAN Problem_Set categories, as well as PROMISE_PLANNED_EXCESS.

The PROMISE_PLANNED_EXCESS model has fields that references these models:

Item_Request, Item_Promise, Item_Acceptance, Delivery_Request, Delivery_Promise, Delivery_Acceptance, Operation_Plan.

item_request -- a Item_Request field of model PROMISE_PLANNED_EXCESS
The Item_Request that has this PROMISE_PLANNED_EXCESS Problem.
Properties: Export-Only Field

item_promise -- a Item_Promise field of model PROMISE_PLANNED_EXCESS
The Item_Promise that has this PROMISE_PLANNED_EXCESS Problem. This 'item_promise.delivery_plan' is planned to deliver more 'quantity' than this 'item_promise.quantity' range.
Properties: Export-Only Field

item_acceptance -- a Item_Acceptance field of model PROMISE_PLANNED_EXCESS
The Item_Acceptance for the 'item_request'.
Properties: Export-Only Field

delivery_request -- a Delivery_Request field of model PROMISE_PLANNED_EXCESS
This is simply shorthand for 'item_request.owner'.
Properties: Export-Only Field

delivery_promise -- a Delivery_Promise field of model PROMISE_PLANNED_EXCESS
This is simply shorthand for 'item_promise.owner'.
Properties: Export-Only Field

delivery_acceptance -- a Delivery_Acceptance field of model PROMISE_PLANNED_EXCESS
This is simply shorthand for 'item_acceptance.owner'.
Properties: Export-Only Field

<i>Extensions</i>	<i>ACCEPTANCE_NOT_PLANNED Extension</i>
-------------------	---

request -- a Request field of model PROMISE_PLANNED_EXCESS
This is simply shorthand for 'item_request.delivery_request.owner'.
Properties: Export-Only Field

promise -- a Promise field of model PROMISE_PLANNED_EXCESS
This is simply shorthand for 'item_promise.delivery_promise.owner'.
Properties: Export-Only Field

acceptance -- a Acceptance field of model PROMISE_PLANNED_EXCESS
This is simply shorthand for 'item_acceptance.delivery_acceptance.owner'.
Properties: Export-Only Field

delivery_plan -- a Operation_Plan field of model PROMISE_PLANNED_EXCESS

This is simply shorthand for 'item_promise.delivery_plan'. This 'delivery_plan' is planned to deliver more than 'item_promise.quantity'.
Properties: Export-Only Field

ACCEPTANCE_NOT_PLANNED -- a category extension of model Problem

A ACCEPTANCE_NOT_PLANNED Problem indicates that the 'item_acceptance' was accepted after the 'item_promise' was offered -- and -- the 'delivery_plan' has not been planned to satisfy the 'item_acceptance'; either there is no 'delivery_plan', or the 'delivery_plan' is for the older 'item_promise'. This Problem will not occur if the 'delivery_acceptance' has an infinite 'due Date', the 'item_acceptance' is for a zero 'quantity', or the 'item_promise' was offered after the 'item_acceptance' was issued.

To resolve this Problem, either eliminate or zero out the 'item_acceptance' (unlikely), 'offer a Promise, or plan to satisfy' the 'item_acceptance' which will create and/or replan the 'delivery_plan' to meet the 'item_acceptance'.

This is in the ACCEPTANCE_Problem_Set category, as well as ACCEPTANCE_NOT_PLANNED.

The ACCEPTANCE_NOT_PLANNED model has fields that references these models
Item_Request, Item_Promise, Item_Acceptance, Delivery_Request, Delivery_Promise, Delivery_Acceptance, Operation_Plan.

Extensions	ACCEPTANCE_NOT_PLANNED Extension
------------	----------------------------------

item_request - - - a Item_Request field of model ACCEPTANCE_NOT_PLANNED
The item_Acceptance that has this ACCEPTANCE_NOT_PLANNED Problem. This
item_acceptance.delivery_plan' either is nonexistent or is planned for this
item_acceptance.item_promise', which is older than this item_Acceptance.
Properties: Export-Only Field

item_promise - - - a Item_Promise field of model
ACCEPTANCE_NOT_PLANNED
The item_Promise for the item_request'.
Properties: Export-Only Field

item_acceptance - - - a Item_Acceptance field of model
ACCEPTANCE_NOT_PLANNED
The item_Acceptance for the item_request'.
Properties: Export-Only Field

delivery_request - - - a Delivery_Request field of model
ACCEPTANCE_NOT_PLANNED
This is simply shorthand for item_request.owner'.
Properties: Export-Only Field

delivery_promise - - - a Delivery_Promise field of model
ACCEPTANCE_NOT_PLANNED
This is simply shorthand for item_promise.owner'.
Properties: Export-Only Field

delivery_acceptance - - - a Delivery_Acceptance field of model
ACCEPTANCE_NOT_PLANNED
This is simply shorthand for item_acceptance.owner'.
Properties: Export-Only Field

request - - - a Request field of model ACCEPTANCE_NOT_PLANNED
This is simply shorthand for item_request.delivery_request.owner'.
Properties: Export-Only Field

promise - - - a Promise field of model ACCEPTANCE_NOT_PLANNED
This is simply shorthand for item_promise.delivery_promise.owner'.
Properties: Export-Only Field

Extensions	ACCEPTANCE_PLANNED_LATE Extension
------------	-----------------------------------

Properties: Export-Only Field

delivery_plan - - - a Operation_Plan field of model
ACCEPTANCE_NOT_PLANNED
This is simply shorthand for item_acceptance.delivery_plan'.
Properties: Export-Only Field

ACCEPTANCE_PLANNED_LATE - - - a category extension of model Problem

A ACCEPTANCE_PLANNED_LATE Problem indicates that the 'delivery_plan' has
been planned to satisfy the item_acceptance' but is in fact planned later than the
'delivery_acceptance's due' Dates. This Problem will not occur if the
'delivery_acceptance' has an infinite due' Date.

To resolve this Problem, either set the 'delivery_acceptance.due' to be later (unlikely),
or adjust the 'delivery_plan' such that its end Date is within due'.

This is in both the ACCEPTANCE and ACCEPTANCE_PLAN Problem_Set categories,
as well as ACCEPTANCE_PLANNED_LATE.

The ACCEPTANCE_PLANNED_LATE model has fields that references these models
: Item_Request, Item_Promise, Item_Acceptance, Delivery_Request,
Delivery_Promise, Delivery_Acceptance, Operation_Plan.

item_request - - - a Item_Request field of model
ACCEPTANCE_PLANNED_LATE
The item_Acceptance that has this ACCEPTANCE_PLANNED_LATE Problem. This
item_acceptance.delivery_plan' is planned to end later than this
item_acceptance.owner.due' (the due Date of this item_Acceptance).
Properties: Export-Only Field

item_promise - - - a Item_Promise field of model
ACCEPTANCE_PLANNED_LATE
The item_Promise for the item_request'.
Properties: Export-Only Field

item_acceptance - - - a Item_Acceptance field of model
ACCEPTANCE_PLANNED_LATE
The item_Acceptance for the item_request'.
Properties: Export-Only Field

Extensions	ACCEPTANCE_PLANNED_EARLY Extension
------------	------------------------------------

delivery_request - - - *a Delivery_Request field of model*
ACCEPTANCE_PLANNED_LATE
 This is simply shorthand for 'item_request.owner'. The 'delivery_plan' is planned to end later than this 'delivery_acceptance.due'.
 Properties: Export-Only Field

delivery_promise - - - *a Delivery_Promise field of model*
ACCEPTANCE_PLANNED_LATE
 This is simply shorthand for 'item_promise.owner'.
 Properties: Export-Only Field

delivery_acceptance - - - *a Delivery_Acceptance field of model*
ACCEPTANCE_PLANNED_LATE
 This is simply shorthand for 'item_acceptance.owner'.
 Properties: Export-Only Field

request - - - *a Request field of model* **ACCEPTANCE_PLANNED_LATE**
 This is simply shorthand for 'item_request.delivery_request.owner'.
 Properties: Export-Only Field

promise - - - *a Promise field of model* **ACCEPTANCE_PLANNED_LATE**
 This is simply shorthand for 'item_promise.delivery_promise.owner'.
 Properties: Export-Only Field

acceptance - - - *a Acceptance field of model* **ACCEPTANCE_PLANNED_LATE**
 This is simply shorthand for 'item_acceptance.delivery_acceptance.owner'.
 Properties: Export-Only Field

delivery_plan - - - *a Operation_Plan field of model*
ACCEPTANCE_PLANNED_LATE
 This is simply shorthand for 'item_acceptance.delivery_plan'. This 'delivery_plan' is planned to end later than 'delivery_acceptance.due'.
 Properties: Export-Only Field

ACCEPTANCE_PLANNED_EARLY - - *a category extension of model* Problem
 A **ACCEPTANCE_PLANNED_EARLY** Problem indicates that the 'delivery_plan' has been planned to satisfy the 'delivery_acceptance' but is in fact planned earlier than the 'delivery_acceptance's due' Date. This Problem will not occur if the 'delivery_acceptance' has an infinite 'due' Date.

Extensions	ACCEPTANCE_PLANNED_EARLY Extension
------------	------------------------------------

To resolve this Problem, either set the 'delivery_acceptance.due' to be earlier (unlikely), or adjust the 'delivery_plan' such that its end Date is within 'due'.

This is in both the **ACCEPTANCE** and **ACCEPTANCE_PLAN** Problem Set categories, as well as **ACCEPTANCE_PLANNED_EARLY**.

The **ACCEPTANCE_PLANNED_EARLY** model has fields that references these models:

Item_Request, **Item_Promise**, **Item_Acceptance**, **Delivery_Request**, **Delivery_Promise**, **Delivery_Acceptance**, **Operation_Plan**.

item_request - - - *a Item_Request field of model*
ACCEPTANCE_PLANNED_EARLY
 The **Item_Acceptance** that has this **ACCEPTANCE_PLANNED_EARLY** Problem. This **Item_acceptance.delivery_plan** is planned to end earlier than this **item_acceptance.owner.due** (the due Date of this **Item_Acceptance**).
 Properties: Export-Only Field

item_promise - - - *a Item_Promise field of model*
ACCEPTANCE_PLANNED_EARLY
 The **Item_Promise** for the **item_request**.
 Properties: Export-Only Field

item_acceptance - - - *a Item_Acceptance field of model*
ACCEPTANCE_PLANNED_EARLY
 The **Item_Acceptance** for the **item_request**.
 Properties: Export-Only Field

delivery_request - - - *a Delivery_Request field of model*
ACCEPTANCE_PLANNED_EARLY
 This is simply shorthand for 'item_request.owner'. The 'delivery_plan' is planned to end earlier than this 'delivery_acceptance.due'.
 Properties: Export-Only Field

delivery_promise - - - *a Delivery_Promise field of model*
ACCEPTANCE_PLANNED_EARLY
 This is simply shorthand for 'item_promise.owner'.
 Properties: Export-Only Field

delivery_acceptance - - - a Delivery_Acceptance field of model
ACCEPTANCE_PLANNED_EARLY
This is simply shorthand for 'item_acceptance.owner'.
Properties: Export-Only Field

request - - - a Request field of model ACCEPTANCE_PLANNED_EARLY
This is simply shorthand for 'item_request.delivery_request.owner'.
Properties: Export-Only Field

promise - - - a Promise field of model ACCEPTANCE_PLANNED_EARLY
This is simply shorthand for 'item_promise.delivery_promise.owner'.
Properties: Export-Only Field

acceptance - - - a Acceptance field of model ACCEPTANCE_PLANNED_EARLY
This is simply shorthand for 'item_acceptance.delivery_acceptance.owner'.
Properties: Export-Only Field

delivery_plan - - - a Operation_Plan field of model
ACCEPTANCE_PLANNED_EARLY
This is simply shorthand for 'item_acceptance.delivery_plan'. This 'delivery_plan' is
planned to end earlier than 'delivery_acceptance.due'.
Properties: Export-Only Field

ACCEPTANCE_PLANNED_SHORT - - - a category extension of model Problem

A ACCEPTANCE_PLANNED_SHORT Problem indicates that the 'delivery_plan' has
been planned to satisfy the 'item_acceptance' but is in fact planned for less 'quantity'
than accepted. This Problem will not occur if the 'item_acceptance' has an infinite
'quantity' range.

To resolve this Problem, either reduce 'item_acceptance.quantity' (unlikely), or
increase the Quantity planned by 'item_acceptance.delivery_plan' to be within the
accepted Quantity_Range.

This is in both the ACCEPTANCE and ACCEPTANCE_PLAN Problem Set categories,
as well as ACCEPTANCE_PLANNED_SHORT.

The ACCEPTANCE_PLANNED_SHORT model has fields that references these models:
Item_Request, Item_Promise, Item_Acceptance, Delivery_Request,
Delivery_Promise, Delivery_Acceptance, Operation_Plan.

item_request - - - a Item_Request field of model
ACCEPTANCE_PLANNED_SHORT
The Item_Acceptance that has this ACCEPTANCE_PLANNED_SHORT Problem.
This 'item_acceptance.delivery_plan' is planned to deliver less 'quantity' than this
'item_acceptance.quantity' range.
Properties: Export-Only Field

item_promise - - - a Item_Promise field of model
ACCEPTANCE_PLANNED_SHORT
The Item_Promise for the 'item_request'.
Properties: Export-Only Field

item_acceptance - - - a Item_Acceptance field of model
ACCEPTANCE_PLANNED_SHORT
The Item_Acceptance for the 'item_request'.
Properties: Export-Only Field

delivery_request - - - a Delivery_Request field of model
ACCEPTANCE_PLANNED_SHORT
This is simply shorthand for 'item_request.owner'.
Properties: Export-Only Field

delivery_promise - - - a Delivery_Promise field of model
ACCEPTANCE_PLANNED_SHORT
This is simply shorthand for 'item_promise.owner'.
Properties: Export-Only Field

delivery_acceptance - - - a Delivery_Acceptance field of model
ACCEPTANCE_PLANNED_SHORT
This is simply shorthand for 'item_acceptance.owner'.
Properties: Export-Only Field

request - - - a Request field of model ACCEPTANCE_PLANNED_SHORT
This is simply shorthand for 'item_request.delivery_request.owner'.
Properties: Export-Only Field

promise - - - a Promise field of model ACCEPTANCE_PLANNED_SHORT
This is simply shorthand for 'item_promise.delivery_promise.owner'.
Properties: Export-Only Field

Extensions

ACCEPTANCE_PLANNED_EXCESS Extension

acceptance - - *a* Acceptance field of model ACCEPTANCE_PLANNED_SHORT
This is simply shorthand for item_acceptance.delivery_acceptance.owner.
Properties: Export-Only Field

delivery_plan - - *a* Operation_Plan field of model
ACCEPTANCE_PLANNED_SHORT
This is simply shorthand for item_acceptance.delivery_plan. This delivery_plan is
planned to deliver less than item_acceptance.quantity.
Properties: Export-Only Field

ACCEPTANCE_PLANNED_EXCESS - - *a* category extension of model Problem

A ACCEPTANCE_PLANNED_EXCESS Problem indicates that the delivery_plan
has been planned to satisfy the item_acceptance but is in fact planned for more quan-
tity than accepted. This Problem will not occur if the item_acceptance has an infinite
quantity range.

To resolve this Problem, either increase item_acceptance.quantity (unlikely), or
reduce the Quantity planned by item_acceptance.delivery_plan to be within the
accepted Quantity_Range.

This is in both the ACCEPTANCE and ACCEPTANCE_PLAN Problem_Set cate-
gories, as well as ACCEPTANCE_PLANNED_EXCESS.

The ACCEPTANCE_PLANNED_EXCESS model has fields that references these
models :

Item_Request, Item_Promise, Item_Acceptance, Delivery_Request,
Delivery_Promise, Delivery_Acceptance, Operation_Plan.

item_request - - *a* Item_Request field of model
ACCEPTANCE_PLANNED_EXCESS

The item_Acceptance that has this ACCEPTANCE_PLANNED_EXCESS Problem.
This item_acceptance.delivery_plan is planned to deliver more quantity than this
item_acceptance.quantity range.

Properties: Export-Only Field

item_promise - - *a* Item_Promise field of model
ACCEPTANCE_PLANNED_EXCESS

The item_Promise for the item_request.

Properties: Export-Only Field

Extensions

PLANNED_BEFORE_CURRENT Extension

item_acceptance - - *a* Item_Acceptance field of model
ACCEPTANCE_PLANNED_EXCESS
The item_Acceptance for the item_request.
Properties: Export-Only Field

delivery_request - - *a* Delivery_Request field of model
ACCEPTANCE_PLANNED_EXCESS
This is simply shorthand for item_request.owner.
Properties: Export-Only Field

delivery_promise - - *a* Delivery_Promise field of model
ACCEPTANCE_PLANNED_EXCESS
This is simply shorthand for item_promise.owner.
Properties: Export-Only Field

delivery_acceptance - - *a* Delivery_Acceptance field of model
ACCEPTANCE_PLANNED_EXCESS
This is simply shorthand for item_acceptance.owner.
Properties: Export-Only Field

request - - *a* Request field of model ACCEPTANCE_PLANNED_EXCESS
This is simply shorthand for item_request.delivery_request.owner.
Properties: Export-Only Field

promise - - *a* Promise field of model ACCEPTANCE_PLANNED_EXCESS
This is simply shorthand for item_promise.delivery_promise.owner.
Properties: Export-Only Field

acceptance - - *a* Acceptance field of model ACCEPTANCE_PLANNED_EXCESS
This is simply shorthand for item_acceptance.delivery_acceptance.owner.
Properties: Export-Only Field

delivery_plan - - *a* Operation_Plan field of model
ACCEPTANCE_PLANNED_EXCESS

This is simply shorthand for item_acceptance.delivery_plan. This delivery_plan is
planned to deliver more than item_acceptance.quantity.

Properties: Export-Only Field

PLANNED_BEFORE_CURRENT - - *a* category extension of model Problem

A **PLANNED_BEFORE_CURRENT** Problem indicates that the `operation_plan` has been planned such that it starts before the plan current date. Only the top operation plan of an operation plan tree may have this problem.

The **PLANNED_BEFORE_CURRENT** model has fields that references these models :

`Operation_Plan`.

`operation_plan` - - *a Operation_Plan field of model*
PLANNED_BEFORE_CURRENT
 The `Operation_Plan` with this **PLANNED_BEFORE_CURRENT** Problem.
 Properties: Export-Only Field

UNRELEASED -- a category extension of model Problem

An **UNRELEASED** Problem indicates that the `operation_plan` has been planned such that it starts before the release_fence and is not released. Only the top operation plan of an operation plan tree may have this problem.

The **UNRELEASED** model has fields that references these models :

`Operation_Plan`.

`operation_plan` - - *a Operation_Plan field of model* **UNRELEASED**
 The `Operation_Plan` with this **UNRELEASED** Problem.
 Properties: Export-Only Field

NEEDS_RELEASE -- a category extension of model Problem

A **NEEDS_RELEASE** Problem indicates that the `operation_plan` has been planned such that it starts before the release_soon_fence and has not been released. Only the top operation plan of an operation plan tree may have this problem.

The **NEEDS_RELEASE** model has fields that references these models :

`Operation_Plan`.

`operation_plan` - - *a Operation_Plan field of model* **NEEDS_RELEASE**
 The `Operation_Plan` with this **NEEDS_RELEASE** Problem.
 Properties: Export-Only Field

INCONSISTENT_OPPLAN -- a category extension of model Problem

A **INCONSISTENT_OPPLAN** Problem indicates that an `operation_plan` should have been replanned, but was not, because to do so would have required replanning an operation plan which had `locked_as_planned` set to `true`. Note that the **INCONSISTENT_OPPLAN** problem is associated with the operation plan that could not be replanned; it may or may not have its own `locked_as_planned` set to `true`. This problem arises only when some property of an operation changes (per-unit time, for example) and the necessary replanning was prevented because some of the operation plans that needed to be replanned had `locked_as_planned` set to `true`. An example of when this can happen would be in a routing with three operations. An operation plan for this routing may have the two "outer" sub-operation plans with `locked_as_planned` set to `true`. If the per-unit time of the middle sub-operation plan is changed and the necessary replanning would cause the outer sub-operation plans to be changed, **INCONSISTENT_OPPLAN** problems would be created.

Note that the resolver for the **INCONSISTENT_OPPLAN** problem does nothing: the only way to remove the **INCONSISTENT_OPPLAN** problem is to set `locked_as_planned` to `false` on the operation plan(s) which prevented the replanning. When a process extension is changed and there are associated operation plans with that operation with `locked_as_planned` set to `true`, undoing the change of process extension may leave extraneous **INCONSISTENT_OPPLAN** problems. This is a known limitation in undoing process extension changes in the presence of associated operation_plans with `locked_as_planned` set to `true`. Unlocking those operation plans will eliminate the problems.

The **INCONSISTENT_OPPLAN** model has fields that references these models :

`Operation_Plan`.

`operation_plan` - - *a Operation_Plan field of model* **INCONSISTENT_OPPLAN**
 The `Operation_Plan` with this **INCONSISTENT_OPPLAN** Problem.
 Properties: Export-Only Field

REQUEST_PROMISED_LATE -- a category extension of model Problem

A **REQUEST_PROMISED_LATE** Problem indicates that the `item_promise` has due Dates that are later than the due Dates of its `item_request`. The Promise is for later than requested. To resolve this Problem, either the Request must be made for later (unlikely), or the Promise must be offered for earlier.

This is in the **REQUEST_PROMISE** and **REQUEST_PROMISE_Problem_Sci categories**, as well as **REQUEST_PROMISED_LATE**.

The **REQUEST_PROMISED_LATE** model has fields that references these models :

`Delivery_Request`, `Delivery_Promise`, `Delivery_Acceptance`.

*Extensions***REQUEST_PROMISED_EARLY** Extension

delivery_request -- *a Delivery_Request field of model*

REQUEST_PROMISED_LATE

The Delivery_Request whose delivery_promise has later due Dates.

Properties: Export-Only Field

delivery_promise -- *a Delivery_Promise field of model*

REQUEST_PROMISED_LATE

The Delivery_Promise whose due Dates are later than its 'delivery_request'.

Properties: Export-Only Field

delivery_acceptance -- *a Delivery_Acceptance field of model*

REQUEST_PROMISED_LATE

The Delivery_Acceptance for the 'delivery_request'.

Properties: Export-Only Field

request -- *a Request field of model REQUEST_PROMISED_LATE*

This is simply shorthand for 'delivery_request.owner'.

Properties: Export-Only Field

promise -- *a Promise field of model REQUEST_PROMISED_LATE*

This is simply shorthand for 'delivery_promise.owner'.

Properties: Export-Only Field

acceptance -- *a Acceptance field of model REQUEST_PROMISED_LATE*

This is simply shorthand for 'delivery_acceptance.owner'.

Properties: Export-Only Field

REQUEST_PROMISED_EARLY -- *a category extension of model Problem*

A **REQUEST_PROMISED_EARLY** Problem indicates that the 'delivery_promise' has earlier 'due Dates' than its 'delivery_request'. The Promise is for earlier than requested. To resolve this Problem, either the Request must be made for earlier or the Promise for later.

This is in the **REQUEST_PROMISE**, and **REQUEST_PROMISE_Problem_Set** categories, as well as **REQUEST_PROMISED_EARLY**.

The **REQUEST_PROMISED_EARLY** model has fields that reference these models: **Delivery_Request**, **Delivery_Promise**, **Delivery_Acceptance**.

*Extensions***REQUEST_PROMISED_SHORT** Extension

delivery_request -- *a Delivery_Request field of model*

REQUEST_PROMISED_EARLY

The Delivery_Request whose delivery_promise has earlier due Dates.

Properties: Export-Only Field

delivery_promise -- *a Delivery_Promise field of model*

REQUEST_PROMISED_EARLY

The Delivery_Promise whose due Dates are earlier than its 'delivery_request'.

Properties: Export-Only Field

delivery_acceptance -- *a Delivery_Acceptance field of model*

REQUEST_PROMISED_EARLY

The Delivery_Acceptance for the 'delivery_request'.

Properties: Export-Only Field

request -- *a Request field of model REQUEST_PROMISED_EARLY*

This is simply shorthand for 'delivery_request.owner'.

Properties: Export-Only Field

promise -- *a Promise field of model REQUEST_PROMISED_EARLY*

This is simply shorthand for 'delivery_promise.owner'.

Properties: Export-Only Field

acceptance -- *a Acceptance field of model REQUEST_PROMISED_EARLY*

This is simply shorthand for 'delivery_acceptance.owner'.

Properties: Export-Only Field

REQUEST_PROMISED_SHORT -- *a category extension of model Problem*

A **REQUEST_PROMISED_SHORT** Problem indicates that the 'Item_promise' has less quantity than its 'Item_request'. The Promise is for less than requested. To resolve this Problem, either the Request must be made for less or the Promise for made for more.

This is in the **REQUEST_PROMISE**, and **REQUEST_PROMISE_Problem_Set** categories, as well as **REQUEST_PROMISED_SHORT**.

The **REQUEST_PROMISED_SHORT** model has fields that reference these models: **Item_Request**, **Item_Promise**, **Item_Acceptance**, **Delivery_Request**, **Delivery_Promise**, **Delivery_Acceptance**.

item_request -- *a* Item_Request field of model REQUEST_PROMISED_SHORT
 The item_request whose item_promise has less quantity.
 Properties: Export-Only Field

item_promise -- *a* Item_Promise field of model REQUEST_PROMISED_SHORT
 The item_promise whose quantity is less than its item_request.
 Properties: Export-Only Field

item_acceptance -- *a* Item_Acceptance field of model REQUEST_PROMISED_SHORT
 The item_acceptance for the item_request.
 Properties: Export-Only Field

delivery_request -- *a* Delivery_Request field of model REQUEST_PROMISED_SHORT
 This is simply shorthand for item_request.owner.
 Properties: Export-Only Field

delivery_promise -- *a* Delivery_Promise field of model REQUEST_PROMISED_SHORT
 This is simply shorthand for item_promise.owner.
 Properties: Export-Only Field

delivery_acceptance -- *a* Delivery_Acceptance field of model REQUEST_PROMISED_SHORT
 This is simply shorthand for item_acceptance.owner.
 Properties: Export-Only Field

request -- *a* Request field of model REQUEST_PROMISED_SHORT
 This is simply shorthand for item_request.delivery_request.owner.
 Properties: Export-Only Field

promise -- *a* Promise field of model REQUEST_PROMISED_SHORT
 This is simply shorthand for item_promise.delivery_promise.owner.
 Properties: Export-Only Field

acceptance -- *a* Acceptance field of model REQUEST_PROMISED_SHORT
 This is simply shorthand for item_acceptance.delivery_acceptance.owner.
 Properties: Export-Only Field

REQUEST_PROMISED_EXCESS -- *a* category extension of model Problem

A REQUEST_PROMISED_EXCESS Problem indicates that the item_promise has more quantity than its item_request. The Promise is for more than requested. To resolve this Problem, either the Request must be made for more or the Promise for made for less.

This is in the REQUEST_PROMISE and REQUEST_PROMISE Problem_Sets categories, as well as REQUEST_PROMISED_EXCESS.

The REQUEST_PROMISED_EXCESS model has fields that references these models

item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance.

item_request -- *a* Item_Request field of model REQUEST_PROMISED_EXCESS
 The item_request whose item_promise has excess quantity.
 Properties: Export-Only Field

item_promise -- *a* Item_Promise field of model REQUEST_PROMISED_EXCESS
 The item_promise whose quantity is more than its item_request.
 Properties: Export-Only Field

item_acceptance -- *a* Item_Acceptance field of model REQUEST_PROMISED_EXCESS
 The item_acceptance for the item_request.
 Properties: Export-Only Field

delivery_request -- *a* Delivery_Request field of model REQUEST_PROMISED_EXCESS
 This is simply shorthand for item_request.owner.
 Properties: Export-Only Field

delivery_promise -- *a* Delivery_Promise field of model REQUEST_PROMISED_EXCESS
 This is simply shorthand for item_promise.owner.
 Properties: Export-Only Field

Extensions	ITEM_PROMISE_OVERPRICED Extension
------------	-----------------------------------

delivery_acceptance -- *a Delivery_Acceptance field of model REQUEST_PROMISED_EXCESS*

This is simply shorthand for 'item_acceptance.owner'.

Properties: Export-Only Field

request -- *a Request field of model REQUEST_PROMISED_EXCESS*
This is simply shorthand for 'item_request.delivery_request.owner'.

Properties: Export-Only Field

promise -- *a Promise field of model REQUEST_PROMISED_EXCESS*
This is simply shorthand for 'item_promise.delivery_promise.owner'.

Properties: Export-Only Field

acceptance -- *a Acceptance field of model REQUEST_PROMISED_EXCESS*
This is simply shorthand for 'item_acceptance.delivery_acceptance.owner'.

Properties: Export-Only Field

ITEM_PROMISE_OVERPRICED -- a category extension of model Problem

An ITEM_PROMISE_OVERPRICED Problem indicates that the 'item_promise' has a higher 'price' than its 'item_request's 'max_price'. The promise is more expensive than requested. To resolve this Problem, either the Request must be made for more or the Promise for made for less. The latter is often done by setting 'discount'.

This is in the REQUEST_PROMISE and REQUEST_PROMISE_Problem_Set categories, as well as ITEM_PROMISE_OVERPRICED.

The ITEM_PROMISE_OVERPRICED model has fields that references these models:
Item_Request, Item_Promise, Item_Acceptance, Delivery_Request,
Delivery_Promise, Delivery_Acceptance.

item_request -- *a Item_Request field of model ITEM_PROMISE_OVERPRICED*

The Item_Request whose 'item_promise' has excess 'quantity'.

Properties: Export-Only Field

item_promise -- *a Item_Promise field of model ITEM_PROMISE_OVERPRICED*

The Item_Promise whose 'quantity' is more than its 'item_request'.

Properties: Export-Only Field

Extensions	DELIVERY_PROMISE_OVERPRICED Extension
------------	---------------------------------------

item_acceptance -- *a Item_Acceptance field of model ITEM_PROMISE_OVERPRICED*

This is simply shorthand for 'item_request'.

Properties: Export-Only Field

delivery_request -- *a Delivery_Request field of model ITEM_PROMISE_OVERPRICED*
This is simply shorthand for 'item_request.owner'.

Properties: Export-Only Field

delivery_promise -- *a Delivery_Promise field of model ITEM_PROMISE_OVERPRICED*
This is simply shorthand for 'item_promise.owner'.

Properties: Export-Only Field

delivery_acceptance -- *a Delivery_Acceptance field of model ITEM_PROMISE_OVERPRICED*
This is simply shorthand for 'item_acceptance.owner'.

Properties: Export-Only Field

request -- *a Request field of model ITEM_PROMISE_OVERPRICED*
This is simply shorthand for 'item_request.delivery_request.owner'.

Properties: Export-Only Field

promise -- *a Promise field of model ITEM_PROMISE_OVERPRICED*
This is simply shorthand for 'item_promise.delivery_promise.owner'.

Properties: Export-Only Field

acceptance -- *a Acceptance field of model ITEM_PROMISE_OVERPRICED*
This is simply shorthand for 'item_acceptance.delivery_acceptance.owner'.

Properties: Export-Only Field

DELIVERY_PROMISE_OVERPRICED -- a category extension of model Problem

A DELIVERY_PROMISE_OVERPRICED Problem indicates that the 'delivery_promise' has a higher 'delivery_price' than its 'delivery_request's 'max_price'. The promise is more expensive than requested. To resolve this Problem, either the Request must be made for more or the Promise for made for less. The latter is often done by setting 'delivery_discount'.

This is in the REQUEST_PROMISE, and REQUEST_PROMISE Problem_Set categories, as well as DELIVERY_PROMISE_OVERPRICED.

The DELIVERY_PROMISE_OVERPRICED model has fields that references these models :

Delivery_Request, Delivery_Promise, Delivery_Acceptance.

delivery_request - - a Delivery_Request field of model
DELIVERY_PROMISE_OVERPRICED

The Delivery_Request whose 'delivery_promise' has excess price.

Properties: Export-Only Field

delivery_promise - - a Delivery_Promise field of model
DELIVERY_PROMISE_OVERPRICED

The Delivery_Promise whose price is more than its 'delivery_request.max_price'.

Properties: Export-Only Field

delivery_acceptance - - a Delivery_Acceptance field of model
DELIVERY_PROMISE_OVERPRICED

The Delivery_Acceptance for the 'delivery_request'.

Properties: Export-Only Field

request - - a Request field of model DELIVERY_PROMISE_OVERPRICED

This is simply shorthand for 'delivery_request.owner'.

Properties: Export-Only Field

promise - - a Promise field of model DELIVERY_PROMISE_OVERPRICED
This is simply shorthand for 'delivery_promise.owner'.

Properties: Export-Only Field

acceptance - - a Acceptance field of model
DELIVERY_PROMISE_OVERPRICED

This is simply shorthand for 'delivery_acceptance.owner'.

Properties: Export-Only Field

PROMISE_NOT_OFFERED - - a category extension of model Problem

A PROMISE_NOT_OFFERED Problem indicates that a Request has been issued, but the Promise has not been offered. More precisely, 'promise_date_offered' is before 'request_date_issued'. To resolve, either the Promise field 'date_offered' must be set to 'date_issued' or later.

This is in the REQUEST and PROMISE Problem_Set categories, as well as PROMISE_NOT_OFFERED.

The PROMISE_NOT_OFFERED model has fields that references these models :

Request, Promise, Acceptance.

request - - a Request field of model PROMISE_NOT_OFFERED
The Request with this PROMISE_NOT_OFFERED Problem.

Properties: Export-Only Field

promise - - a Promise field of model PROMISE_NOT_OFFERED
The Promise for the 'request'.

Properties: Export-Only Field

acceptance - - a Acceptance field of model PROMISE_NOT_OFFERED
The Acceptance for the 'request'.

Properties: Export-Only Field

PROMISE_NOT_ACCEPTED - - a category extension of model Problem

A PROMISE_NOT_ACCEPTED Problem indicates that a Promise has been offered but not yet been accepted. More precisely, 'date_accepted' is before 'date_offered', which is at or after 'date_issued'. To resolve, the field 'date_accepted' must be set to 'date_offered' or later.

This is in the REQUEST and PROMISE Problem_Set categories, as well as PROMISE_NOT_ACCEPTED.

The PROMISE_NOT_ACCEPTED model has fields that references these models :
Request, Promise, Acceptance.

request - - a Request field of model PROMISE_NOT_ACCEPTED
The Request that has this PROMISE_NOT_ACCEPTED Problem.

Properties: Export-Only Field

promise - - a Promise field of model PROMISE_NOT_ACCEPTED

The Promise with this PROMISE_NOT_ACCEPTED Problem.

Properties: Export-Only Field

Extensions	ACCEPTANCE_INCONSISTENT Extension
------------	-----------------------------------

acceptance -- *a* Acceptance field of model **PROMISE_NOT_ACCEPTED**
 The Acceptance for the request.
 Properties: Export-Only Field

ACCEPTANCE_INCONSISTENT -- *a* category extension of model Problem

A **ACCEPTANCE_INCONSISTENT** Problem indicates that a Promise has been accepted but the acceptance date or quantity is different from what had been promised.
 This is in the **REQUEST** and **PROMISE** Problem_Set categories, as well as **ACCEPTANCE_INCONSISTENT**.

The **ACCEPTANCE_INCONSISTENT** model has fields that references these models:
 Request, Promise, Acceptance.

request -- *a* Request field of model **ACCEPTANCE_INCONSISTENT**
 The Request that has this **ACCEPTANCE_INCONSISTENT** Problem.
 Properties: Export-Only Field

promise -- *a* Promise field of model **ACCEPTANCE_INCONSISTENT**
 The Promise for the request.
 Properties: Export-Only Field

acceptance -- *a* Acceptance field of model **ACCEPTANCE_INCONSISTENT**
 The Acceptance with this **ACCEPTANCE_INCONSISTENT** Problem.
 Properties: Export-Only Field

REQUEST_QUEUED -- *a* category extension of model Problem

A **REQUEST_QUEUED** Problem indicates that a Request has been answered unsatisfactorily, and then 'queued' by the requestor for later reconsideration. More precisely, 'date_queued' is at or after 'date_offered', which is at or after 'date_issued'. To resolve, 'date_issued' or the Promise field 'date_offered' must be set to at or after 'date_queued'.

This is in the **REQUEST** Problem_Set category, as well as **REQUEST_QUEUED**.

The **REQUEST_QUEUED** model has fields that references these models:
 Request, Promise, Acceptance.

request -- *a* Request field of model **REQUEST_QUEUED**
 The Request with this **REQUEST_QUEUED** Problem.

Extensions	DELIVERY_REQUEST_NOT_COORDINATED Extension
------------	--

Properties: Export-Only Field
 promise -- *a* Promise field of model **REQUEST_QUEUED**
 The Promise for the request.
 Properties: Export-Only Field

acceptance -- *a* Acceptance field of model **REQUEST_QUEUED**
 The Acceptance for the request.
 Properties: Export-Only Field

DELIVERY_REQUEST_NOT_COORDINATED -- *a* category extension of model Problem

A **DELIVERY_REQUEST_NOT_COORDINATED** Problem indicates that the items of a **Delivery_Request** are not all scheduled for delivery on the same date.

This type of problem is resolved as follows. First, flip a coin to decide whether to correct earliness/lateness. If the coin toss indicates correcting earliness/lateness, then move all early line item deliveries out to the start of the delivery's due period, and move all late line item deliveries in to the end of that period. Whether correcting earliness/lateness or not, proceed by deciding whether to coordinate line item deliveries by either moving in or moving out. If moving in, then move all line item deliveries to the date of the earliest line item delivery. If moving out, then move all line item deliveries to the date of the latest delivery.

The **DELIVERY_REQUEST_NOT_COORDINATED** model has fields that references these models:
 Delivery_Request, Delivery_Promise, Delivery_Acceptance.

delivery_request -- *a* Delivery_Request field of model **DELIVERY_REQUEST_NOT_COORDINATED**
 The **Delivery_Request** with this **DELIVERY_REQUEST_NOT_COORDINATED** Problem.
 Properties: Export-Only Field

delivery_promise -- *a* Delivery_Promise field of model **DELIVERY_REQUEST_NOT_COORDINATED**
 The **Delivery_Promise** for the 'delivery_request'.
 Properties: Export-Only Field

Extensions	DELIVERY_PROMISE_NOT_COORDINATED Extension
------------	--

delivery_acceptance -- *a Delivery_Acceptance field of model*
DELIVERY_REQUEST_NOT_COORDINATED
The Delivery_Acceptance for the delivery_request.
Properties: Export-Only Field

request -- *a Request field of model*
DELIVERY_REQUEST_NOT_COORDINATED
This is simply shorthand for delivery_request.owner.
Properties: Export-Only Field

promise -- *a Promise field of model*
DELIVERY_REQUEST_NOT_COORDINATED
This is simply shorthand for delivery_promise.owner.
Properties: Export-Only Field

acceptance -- *a Acceptance field of model*
DELIVERY_REQUEST_NOT_COORDINATED
This is simply shorthand for delivery_acceptance.owner.
Properties: Export-Only Field

DELIVERY_PROMISE_NOT_COORDINATED -- *a category extension of model Problem*

A **DELIVERY_PROMISE_NOT_COORDINATED** Problem indicates that the items of a Delivery_Promise are not all scheduled for delivery on the same date.

For an explanation of how this problem is resolved, see the **DELIVERY_REQUEST_NOT_COORDINATED** problem.

The **DELIVERY_PROMISE_NOT_COORDINATED** model has fields that references these models :

Delivery_Request, Delivery_Promise, Delivery_Acceptance.

delivery_request -- *a Delivery_Request field of model*
DELIVERY_PROMISE_NOT_COORDINATED
The Delivery_Request that has this **DELIVERY_NOT_COORDINATED** Problem.
Properties: Export-Only Field

delivery_promise -- *a Delivery_Promise field of model*
DELIVERY_PROMISE_NOT_COORDINATED
The Delivery_Promise with this **DELIVERY_PROMISE_NOT_COORDINATED** Problem.

Extensions	DELIVERY_ACCEPTANCE_NOT_COORDINATED Extension
------------	---

Properties: Export-Only Field

delivery_acceptance -- *a Delivery_Acceptance field of model*
DELIVERY_PROMISE_NOT_COORDINATED
The Delivery_Acceptance for the delivery_request.
Properties: Export-Only Field

request -- *a Request field of model*
DELIVERY_PROMISE_NOT_COORDINATED
This is simply shorthand for delivery_request.owner.
Properties: Export-Only Field

promise -- *a Promise field of model*
DELIVERY_PROMISE_NOT_COORDINATED
This is simply shorthand for delivery_promise.owner.
Properties: Export-Only Field

acceptance -- *a Acceptance field of model*
DELIVERY_PROMISE_NOT_COORDINATED
This is simply shorthand for delivery_acceptance.owner.
Properties: Export-Only Field

DELIVERY_ACCEPTANCE_NOT_COORDINATED -- *a category extension of model Problem*

A **DELIVERY_ACCEPTANCE_NOT_COORDINATED** Problem indicates that the items of a Delivery_Acceptance are not all scheduled for delivery on the same date.

For an explanation of how this problem is resolved, see the **DELIVERY_REQUEST_NOT_COORDINATED** problem.

The **DELIVERY_ACCEPTANCE_NOT_COORDINATED** model has fields that references these models :

Delivery_Request, Delivery_Promise, Delivery_Acceptance.

delivery_request -- *a Delivery_Request field of model*
DELIVERY_ACCEPTANCE_NOT_COORDINATED
The Delivery_Request that has this **DELIVERY_ACCEPTANCE_NOT_COORDINATED** Problem.
Properties: Export-Only Field

<i>Extensions</i>	<i>NEGATIVE_ATP Extension</i>
-------------------	-------------------------------

delivery_promise -- a Delivery_Promise field of model
 DELIVERY_ACCEPTANCE_NOT_COORDINATED
 The Delivery_Promise for the delivery_request.
 Properties: Export-Only Field

delivery_acceptance -- a Delivery_Acceptance field of model
 DELIVERY_ACCEPTANCE_NOT_COORDINATED
 The Delivery_Acceptance with this
 DELIVERY_ACCEPTANCE_NOT_COORDINATED Problem.
 Properties: Export-Only Field

request -- a Request field of model
 DELIVERY_ACCEPTANCE_NOT_COORDINATED
 This is simply shorthand for delivery_request.owner.
 Properties: Export-Only Field

promise -- a Promise field of model
 DELIVERY_ACCEPTANCE_NOT_COORDINATED
 This is simply shorthand for delivery_promise.owner.
 Properties: Export-Only Field

acceptance -- a Acceptance field of model
 DELIVERY_ACCEPTANCE_NOT_COORDINATED
 This is simply shorthand for delivery_acceptance.owner.
 Properties: Export-Only Field

NEGATIVE_ATP -- a category extension of model Problem

A NEGATIVE_ATP Problem indicates that the available_to_promise quantity is negative for a particular Forecast_Entry. This occurs when more Product is consumed from the Forecast_Entry than was allocated. This situation may arise when one of two things happens: (1) the allocated quantity is reduced, or (2) the consumption increases. Note, however, that not all of this consumption need occur in the Forecast_Entry which has a problem. Some of the available_to_promise may have been consumed from a later Forecast_Entry (if the Product for the Forecast_Entry specifies a consume_earlier which allows this case.)

There is no automated resolution of this Problem. The allocation will need to be increased, or the actual promises that are consuming the allocation will need to be reduced.

The NEGATIVE_ATP model has fields that references these models :

<i>Extensions</i>	<i>NEGATIVE_PLANNED_ATP Extension</i>
-------------------	---------------------------------------

Forecast_Entry.
 forecast_entry -- a Forecast_Entry field of model NEGATIVE_ATP
 The Forecast_Entry in which this problem occurred.
 Properties: Export-Only Field

NEGATIVE_PLANNED_ATP -- a category extension of model Problem

A NEGATIVE_PLANNED_ATP Problem indicates that the planned_available quantity is negative for a particular Forecast_Entry. This occurs when more Product is consumed from the Forecast_Entry than was planned. This situation may arise when one of two things happens: (1) the planned quantity is reduced, or (2) the consumption increases. Note, however, that not all of this consumption need occur in the Forecast_Entry which has a problem. Some of the planned_available may have been consumed from a later Forecast_Entry (if the Product for the Forecast_Entry specifies a consume_earlier which allows this case.)

There is no automated resolution of this Problem. The allocation will need to be increased, or the actual promises that are consuming the allocation will need to be reduced.

The NEGATIVE_PLANNED_ATP model has fields that references these models :
 Forecast_Entry.

forecast_entry -- a Forecast_Entry field of model NEGATIVE_PLANNED_ATP
 The Forecast_Entry in which this problem occurred.
 Properties: Export-Only Field

OVER_COMMITTED -- a category extension of model Problem

An OVER_COMMITTED Problem indicates that the committed quantity is greater than the forecasted quantity for a particular Forecast_Entry. This occurs in one of two ways: (1) the committed quantity is edited to be greater than the forecasted quantity, or (2) the forecasted is edited to be less than the committed quantity. These situations violate the forecast limit imposed by the forecasted field, and thus, produce this problem.

This problem is resolved by reducing the Forecast_Entry 'committed' field to be no greater than the 'forecasted' field. This effectively honors the limit set by the 'forecasted' field. If no limit is desired, the 'forecasted' field should be set to 'oo'.

The OVER_COMMITTED model has fields that references these models :

<i>Extensions</i>	<i>OVER_CONSUMED Extension</i>
-------------------	--------------------------------

Forecast_Entry.

forecast_entry - - *a Forecast_Entry field of model OVER_COMMITTED*
The Forecast_Entry in which this problem occurred.
Properties: Export-Only Field

OVER_CONSUMED - - *a category extension of model Problem*

An OVER_CONSUMED Problem indicates that the 'consumed' quantity is greater than the 'committed' quantity for a particular Forecast_Entry. This occurs in one of two ways: (1) enough actual requests are promised to make the 'consumed' quantity greater than the 'committed' quantity, or (2) the 'committed' is edited to be less than the 'consumed' quantity.

This problem is resolved by increasing the Forecast_Entry 'committed' field to be equal to the 'consumed' field.

The OVER_CONSUMED model has fields that references these models :

Forecast_Entry.

forecast_entry - - *a Forecast_Entry field of model OVER_CONSUMED*

The Forecast_Entry in which this problem occurred.

Properties: Export-Only Field

UNALLOCATED_FORECAST - - *a category extension of model Problem*

An UNALLOCATED_FORECAST Problem indicates that 'date_committed' is later than 'date_allocated' for a particular Forecast_Entry. This occurs in two ways: (1) the 'committed' is edited after the associated forecast requests have been planned and promised (i.e. allocated), or (2) the 'date_committed' field was manually edited to a value later than the 'date_allocated' field.

This problem is resolved by planning and promising the forecast request(s) associated with this problem's 'forecast_entry'. This action causes the 'date_allocated' field to be set later than the 'date_committed' thereby eliminating this problem.

The UNALLOCATED_FORECAST model has fields that references these models :
Forecast_Entry.

forecast_entry - - *a Forecast_Entry field of model UNALLOCATED_FORECAST*
The Forecast_Entry in which this problem occurred.

Properties: Export-Only Field

<i>Extensions</i>	<i>SUPPLY_PLANNED_LATE Extension</i>
-------------------	--------------------------------------

SUPPLY_PLANNED_LATE - - *a category extension of model Problem*

A SUPPLY_PLANNED_LATE Problem indicates that the 'delivery_plan' has been planned to satisfy the 'item_request' but is in fact planned later than the 'receiving_plan's start date. This Problem generally corresponds to a REQUEST_PLANNED_LATE Problem at the supplying Site.

To resolve this Problem, either resolve the REQUEST_PLANNED_LATE Problem at the supplying Site, or adjust the 'receiving_plan' such that its start Date matches the supplying Site's 'delivery_plan's end Date.

This is in both the SUPPLY and SUPPLY_PLAN Problem_Sets categories, as well as SUPPLY_PLANNED_LATE.

The SUPPLY_PLANNED_LATE model has fields that references these models :

Operation_Plan, Item_Request, Item_Promise, Item_Acceptance,
Delivery_Request, Delivery_Promise, Delivery_Acceptance.

receiving_plan - - *a Operation_Plan field of model SUPPLY_PLANNED_LATE*

This is the Operation_Plan that will receive the delivery at this Site. Often, this is the Operation_Plan that issued the 'item_request'.

Properties: Export-Only Field

item_request - - *a Item_Request field of model SUPPLY_PLANNED_LATE*

The Item_Request that has this SUPPLY_PLANNED_LATE Problem. This item_request:delivery_plan is planned to end later than this item_request:owner:due (the due Date of this Item_Request).

Properties: Export-Only Field

item_promise - - *a Item_Promise field of model SUPPLY_PLANNED_LATE*

The Item_Promise for the 'item_request'.

Properties: Export-Only Field

item_acceptance - - *a Item_Acceptance field of model SUPPLY_PLANNED_LATE*

The Item_Acceptance for the 'item_request'.

Properties: Export-Only Field

Extensions	SUPPLY_PLANNED_EARLY Extension
------------	--------------------------------

delivery_request - - a Delivery_Request field of model SUPPLY_PLANNED_LATE

This is simply shorthand for 'item_request.owner'. The 'delivery_plan' is planned to end later than this 'delivery_request.due'.
Properties: Export-Only Field

delivery_promise - - a Delivery_Promise field of model SUPPLY_PLANNED_LATE
This is simply shorthand for 'item_promise.owner'.
Properties: Export-Only Field

delivery_acceptance - - a Delivery_Acceptance field of model SUPPLY_PLANNED_LATE
This is simply shorthand for 'item_acceptance.owner'.
Properties: Export-Only Field

request - - a Request field of model SUPPLY_PLANNED_LATE
This is simply shorthand for 'item_request.delivery_request.owner'.
Properties: Export-Only Field

promise - - a Promise field of model SUPPLY_PLANNED_LATE
This is simply shorthand for 'item_promise.delivery_promise.owner'.
Properties: Export-Only Field

acceptance - - a Acceptance field of model SUPPLY_PLANNED_LATE
This is simply shorthand for 'item_acceptance.delivery_acceptance.owner'.
Properties: Export-Only Field

delivery_plan - - a Operation_Plan field of model SUPPLY_PLANNED_LATE
This is simply shorthand for 'item_request.delivery_plan'. This 'delivery_plan' is planned to end later than 'delivery_request.due'.
Properties: Export-Only Field

SUPPLY_PLANNED_EARLY - - a category extension of model Problem

A SUPPLY_PLANNED_EARLY Problem indicates that the 'delivery_plan' has been planned to satisfy the 'delivery_request' but is in fact planned earlier than the receiving plan's start Date. This Problem generally corresponds to a REQUEST_PLANNED_EARLY Problem at the supplying Site.

Extensions	SUPPLY_PLANNED_EARLY Extension
------------	--------------------------------

To resolve this Problem, either resolve the REQUEST_PLANNED_EARLY Problem at the supplying Site, or adjust the 'receiving_plan' such that its start Date matches the supplying Site's 'delivery_plan's end Date.

This is in both the SUPPLY_PLANNED_EARLY and SUPPLY_PLAN Problem Set categories, as well as SUPPLY_PLANNED_EARLY.

The SUPPLY_PLANNED_EARLY model has fields that references these models :
Operation_Plan, Item_Request, Item_Promise, Item_Acceptance,
Delivery_Request, Delivery_Promise, Delivery_Acceptance.

receiving_plan - - a Operation_Plan field of model SUPPLY_PLANNED_EARLY
This is the Operation_Plan that will receive the delivery at this Site. Often, this is the Operation_Plan that issued the 'item_request'.
Properties: Export-Only Field

item_request - - a Item_Request field of model SUPPLY_PLANNED_EARLY
The Item_Request that has this SUPPLY_PLANNED_EARLY Problem. This 'item_request.delivery_plan' is planned to end earlier than this 'item_request.owner.due' (the due Date of this Item_Request).
Properties: Export-Only Field

item_promise - - a Item_Promise field of model SUPPLY_PLANNED_EARLY
The Item_Promise for the 'item_request'.
Properties: Export-Only Field

item_acceptance - - a Item_Acceptance field of model SUPPLY_PLANNED_EARLY
The Item_Acceptance for the 'item_request'.
Properties: Export-Only Field

delivery_request - - a Delivery_Request field of model SUPPLY_PLANNED_EARLY
This is simply shorthand for 'item_request.owner'. The 'delivery_plan' is planned to end earlier than this 'delivery_request.due'.
Properties: Export-Only Field

delivery_promise - - a Delivery_Promise field of model SUPPLY_PLANNED_EARLY
This is simply shorthand for 'item_promise.owner'.
Properties: Export-Only Field

Extensions	SUPPLY_PLANNED_SHORT Extension
------------	--------------------------------

delivery_acceptance - - - *a Delivery_Acceptance field of model SUPPLY_PLANNED_EARLY*
This is simply shorthand for 'item_acceptance.owner'.
Properties: Export-Only Field

request - - - *a Request field of model SUPPLY_PLANNED_EARLY*
This is simply shorthand for 'item_request.delivery_request.owner'.
Properties: Export-Only Field

promise - - - *a Promise field of model SUPPLY_PLANNED_EARLY*
This is simply shorthand for 'item_promise.delivery_promise.owner'.
Properties: Export-Only Field

acceptance - - - *a Acceptance field of model SUPPLY_PLANNED_EARLY*
This is simply shorthand for 'item_acceptance.delivery_acceptance.owner'.
Properties: Export-Only Field

delivery_plan - - - *a Operation_Plan field of model SUPPLY_PLANNED_EARLY*
This is simply shorthand for 'item_request.delivery_plan'. This 'delivery_plan' is planned to end earlier than 'delivery_request.due'.
Properties: Export-Only Field

SUPPLY_PLANNED_SHORT - - - *a category extension of model Problem*

A SUPPLY_PLANNED_SHORT Problem indicates that the 'delivery_plan' has been planned to satisfy the 'item_request' but is in fact planned for less 'quantity' than the 'receiving_plan'. This Problem generally corresponds to a REQUEST_PLANNED_SHORT Problem at the supplying Site.

To resolve this Problem, either resolve the REQUEST_PLANNED_SHORT Problem at the supplying Site, or reduce the 'receiving_plan' Quantity such that it matches the supplying Site's 'delivery_plan's Quantity.

This is in both the SUPPLY and SUPPLY_PLAN Problem_Set categories, as well as SUPPLY_PLANNED_SHORT.

The SUPPLY_PLANNED_SHORT model has fields that references these models :
Operation_Plan, Item_Request, Item_Promise, Item_Acceptance,
Delivery_Request, Delivery_Promise, Delivery_Acceptance.

Extensions	SUPPLY_PLANNED_SHORT Extension
------------	--------------------------------

receiving_plan - - - *a Operation_Plan field of model SUPPLY_PLANNED_SHORT*
This is the Operation_Plan that will receive the delivery at this Site. Often, this is the Operation_Plan that issued the 'item_request'.
Properties: Export-Only Field

item_request - - - *a Item_Request field of model SUPPLY_PLANNED_SHORT*
The Item_Request that has this SUPPLY_PLANNED_SHORT Problem. This 'item_request.delivery_plan' is planned to deliver less 'quantity' than this 'item_request.quantity' range.
Properties: Export-Only Field

item_promise - - - *a Item_Promise field of model SUPPLY_PLANNED_SHORT*
The Item_Promise for the 'item_request'.
Properties: Export-Only Field

item_acceptance - - - *a Item_Acceptance field of model SUPPLY_PLANNED_SHORT*
The Item_Acceptance for the 'item_request'.
Properties: Export-Only Field

delivery_request - - - *a Delivery_Request field of model SUPPLY_PLANNED_SHORT*
This is simply shorthand for 'item_request.owner'.
Properties: Export-Only Field

delivery_promise - - - *a Delivery_Promise field of model SUPPLY_PLANNED_SHORT*
This is simply shorthand for 'item_promise.owner'.
Properties: Export-Only Field

delivery_acceptance - - - *a Delivery_Acceptance field of model SUPPLY_PLANNED_SHORT*
This is simply shorthand for 'item_acceptance.owner'.
Properties: Export-Only Field

request - - - *a Request field of model SUPPLY_PLANNED_SHORT*
This is simply shorthand for 'item_request.delivery_request.owner'.
Properties: Export-Only Field

promise - - - *a Promise field of model SUPPLY_PLANNED_SHORT*
This is simply shorthand for 'item_promise.delivery_promise.owner'.

<i>Extensions</i>	<i>SUPPLY_PLANNED_EXCESS Extension</i>
-------------------	--

Properties: Export-Only Field

acceptance -- *a Acceptance field of model SUPPLY_PLANNED_SHORT*

This is simply shorthand for 'item_acceptance.delivery_acceptance.owner'.

Properties: Export-Only Field

delivery_plan -- *a Operation_Plan field of model SUPPLY_PLANNED_SHORT*

This is simply shorthand for 'item_request.delivery_plan'. This 'delivery_plan' is planned to deliver less than 'item_request.quantity'.

Properties: Export-Only Field

SUPPLY_PLANNED_EXCESS -- *a category extension of model Problem*

A SUPPLY_PLANNED_EXCESS Problem indicates that the 'delivery_plan' has been planned to satisfy the 'item_request' but is in fact planned for more quantity than the 'receiving_plan'. This Problem generally corresponds to a REQUEST_PLANNED_EXCESS Problem at the supplying Site.

To resolve this Problem, either resolve the REQUEST_PLANNED_EXCESS Problem at the supplying Site, or increase the 'receiving_plan' Quantity such that it matches the supplying Site's 'delivery_plan's Quantity.

This is in both the SUPPLY and SUPPLY_PLAN Problem_Sets categories, as well as SUPPLY_PLANNED_EXCESS.

The SUPPLY_PLANNED_EXCESS model has fields that references these models :
Operation_Plan, Item_Request, Item_Promise, Item_Acceptance,
Delivery_Request, Delivery_Promise, Delivery_Acceptance.

receiving_plan -- *a Operation_Plan field of model*

SUPPLY_PLANNED_EXCESS

This is the Operation_Plan that will receive the delivery at this Site. Often, this is the Operation_Plan that issued the 'item_request'.

Properties: Export-Only Field

item_request -- *a Item_Request field of model SUPPLY_PLANNED_EXCESS*

The Item_Request that has this SUPPLY_PLANNED_EXCESS Problem. This

'item_request.delivery_plan' is planned to deliver more quantity than this 'item_request.quantity' range.

Properties: Export-Only Field

<i>Extensions</i>	<i>SUPPLY_PLANNED_EXCESS Extension</i>
-------------------	--

item_promise -- *a Item_Promise field of model SUPPLY_PLANNED_EXCESS*
The Item_Promise for the 'item_request'.

Properties: Export-Only Field

item_acceptance -- *a Item_Acceptance field of model*
SUPPLY_PLANNED_EXCESS

The Item_Acceptance for the 'item_request'.

Properties: Export-Only Field

delivery_request -- *a Delivery_Request field of model*
SUPPLY_PLANNED_EXCESS

This is simply shorthand for 'item_request.owner'.

Properties: Export-Only Field

delivery_promise -- *a Delivery_Promise field of model*
SUPPLY_PLANNED_EXCESS

This is simply shorthand for 'item_promise.owner'.

Properties: Export-Only Field

delivery_acceptance -- *a Delivery_Acceptance field of model*
SUPPLY_PLANNED_EXCESS

This is simply shorthand for 'item_acceptance.owner'.

Properties: Export-Only Field

request -- *a Request field of model SUPPLY_PLANNED_EXCESS*

This is simply shorthand for 'item_request.delivery_request.owner'.

Properties: Export-Only Field

promise -- *a Promise field of model SUPPLY_PLANNED_EXCESS*

This is simply shorthand for 'item_promise.delivery_promise.owner'.

Properties: Export-Only Field

acceptance -- *a Acceptance field of model SUPPLY_PLANNED_EXCESS*

This is simply shorthand for 'item_acceptance.delivery_acceptance.owner'.

Properties: Export-Only Field

delivery_plan -- *a Operation_Plan field of model SUPPLY_PLANNED_EXCESS*

This is simply shorthand for 'item_request.delivery_plan'. This 'delivery_plan' is planned to deliver more than 'item_request.quantity'.

Properties: Export-Only Field

<i>Extensions</i>	<i>SUPPLY_PROMISED_LATE Extension</i>
SUPPLY_PROMISED_LATE -- a category extension of model Problem	
<p>A SUPPLY_PROMISED_LATE Problem indicates that the <i>Item_Promise</i> has 'due' Dates that are later than the start Date of the <i>receiving_plan</i>. The <i>Promise</i> is for later than requested. This Problem generally corresponds to a REQUEST_PROMISED_LATE Problem at the supplying Site.</p> <p>To resolve this Problem, either resolve the REQUEST_PROMISED_LATE Problem at the supplying Site, or adjust the <i>receiving_plan</i> such that its start Date matches the supplying Site's <i>delivery_promise</i>'s end Date.</p> <p>This is in the SUPPLY and SUPPLY_PROMISE Problem_Sets categories, as well as SUPPLY_PROMISED_LATE.</p> <p>The SUPPLY_PROMISED_LATE model has fields that references these models :</p> <p><i>Operation_Plan</i>, <i>Item_Request</i>, <i>Item_Promise</i>, <i>Item_Acceptance</i>, <i>Delivery_Request</i>, <i>Delivery_Promise</i>, <i>Delivery_Acceptance</i>.</p> <p><i>receiving_plan</i> -- a <i>Operation_Plan</i> field of model SUPPLY_PROMISED_LATE This is the <i>Operation_Plan</i> that will receive the delivery at this Site. Often, this is the <i>Operation_Plan</i> that issued the <i>Item_request</i>; Properties: <i>Export-Only</i> Field</p> <p><i>Item_request</i> -- a <i>Item_Request</i> field of model SUPPLY_PROMISED_LATE This is the <i>Item_Request</i> which was generated to request the supply Properties: <i>Export-Only</i> Field</p> <p><i>Item_promise</i> -- a <i>Item_Promise</i> field of model SUPPLY_PROMISED_LATE The <i>Item_Promise</i> for the <i>Item_request</i>; Properties: <i>Export-Only</i> Field</p> <p><i>Item_acceptance</i> -- a <i>Item_Acceptance</i> field of model SUPPLY_PROMISED_LATE The <i>Item_Acceptance</i> for the <i>Item_request</i>; Properties: <i>Export-Only</i> Field</p> <p><i>delivery_request</i> -- a <i>Delivery_Request</i> field of model SUPPLY_PROMISED_LATE The <i>Delivery_Request</i> whose <i>delivery_promise</i> has later 'due' Dates than the <i>receiving_plan</i> Properties: <i>Export-Only</i> Field</p>	

<i>Extensions</i>	<i>SUPPLY_PROMISED_EARLY Extension</i>
SUPPLY_PROMISED_EARLY -- a category extension of model Problem	
<p><i>delivery_promise</i> -- a <i>Delivery_Promise</i> field of model SUPPLY_PROMISED_LATE The <i>Delivery_Promise</i> whose 'due' Dates are later than its <i>delivery_request</i>; Properties: <i>Export-Only</i> Field</p> <p><i>delivery_acceptance</i> -- a <i>Delivery_Acceptance</i> field of model SUPPLY_PROMISED_LATE This is simply shorthand for <i>Item_acceptance.owner</i>; Properties: <i>Export-Only</i> Field</p> <p><i>request</i> -- a <i>Request</i> field of model SUPPLY_PROMISED_LATE This is simply shorthand for <i>Item_request.delivery_request.owner</i>; Properties: <i>Export-Only</i> Field</p> <p><i>promise</i> -- a <i>Promise</i> field of model SUPPLY_PROMISED_LATE This is simply shorthand for <i>Item_promise.delivery_promise.owner</i>; Properties: <i>Export-Only</i> Field</p> <p><i>acceptance</i> -- a <i>Acceptance</i> field of model SUPPLY_PROMISED_LATE This is simply shorthand for <i>Item_acceptance.delivery_acceptance.owner</i>; Properties: <i>Export-Only</i> Field</p>	
SUPPLY_PROMISED_EARLY -- a category extension of model Problem	
<p>A SUPPLY_PROMISED_EARLY Problem indicates that the <i>delivery_promise</i> has earlier 'due' Dates than the <i>receiving_plan</i>'s start Date. The <i>Promise</i> is for earlier than requested. This Problem generally corresponds to a REQUEST_PROMISED_EARLY Problem at the supplying Site.</p> <p>To resolve this Problem, either resolve the REQUEST_PROMISED_EARLY Problem at the supplying Site, or adjust the <i>receiving_plan</i> such that its start Date matches the supplying Site's <i>delivery_promise</i>'s end Date.</p> <p>This is in the SUPPLY and SUPPLY_PROMISE Problem_Sets categories, as well as SUPPLY_PROMISED_EARLY.</p> <p>The SUPPLY_PROMISED_EARLY model has fields that references these models :</p> <p><i>Operation_Plan</i>, <i>Item_Request</i>, <i>Item_Promise</i>, <i>Item_Acceptance</i>, <i>Delivery_Request</i>, <i>Delivery_Promise</i>, <i>Delivery_Acceptance</i>.</p>	

Extensions	SUPPLY_PROMISED_EARLY Extension
------------	---------------------------------

receiving_plan -- a Operation_Plan field of model

SUPPLY_PROMISED_EARLY

This is the Operation_Plan that will receive the delivery at this Site. Often, this is the Operation_Plan that issued the item_request.

Properties: Export-Only Field

item_request -- a Item_Request field of model SUPPLY_PROMISED_EARLY

This is the Item_Request which was generated to request the supply

Properties: Export-Only Field

item_promise -- a Item_Promise field of model SUPPLY_PROMISED_EARLY
The Item_Promise for the item_request.

Properties: Export-Only Field

item_acceptance -- a Item_Acceptance field of model
SUPPLY_PROMISED_EARLY

The Item_Acceptance for the item_request.

Properties: Export-Only Field

delivery_request -- a Delivery_Request field of model
SUPPLY_PROMISED_EARLY

The Delivery_Request whose delivery_promise has earlier due Dates.

Properties: Export-Only Field

delivery_promise -- a Delivery_Promise field of model
SUPPLY_PROMISED_EARLY

The Delivery_Promise whose due Dates are earlier than its delivery_request.

Properties: Export-Only Field

delivery_acceptance -- a Delivery_Acceptance field of model
SUPPLY_PROMISED_EARLY

This is simply shorthand for item_acceptance.owner.

Properties: Export-Only Field

request -- a Request field of model SUPPLY_PROMISED_EARLY

This is simply shorthand for item_request.delivery_request.owner.

Properties: Export-Only Field

promise -- a Promise field of model SUPPLY_PROMISED_EARLY

This is simply shorthand for item_promise.delivery_promise.owner.

Properties: Export-Only Field

Extensions	SUPPLY_PROMISED_SHORT Extension
------------	---------------------------------

acceptance -- a Acceptance field of model SUPPLY_PROMISED_EARLY

This is simply shorthand for item_acceptance.delivery_acceptance.owner.

Properties: Export-Only Field

SUPPLY_PROMISED_SHORT -- a category extension of model Problem

A SUPPLY_PROMISED_SHORT Problem indicates that the item_promise has less 'quantity' than the receiving_plan. The Promise is for less than requested. This Problem generally corresponds to a REQUEST_PROMISED_SHORT Problem at the supplying Site.

To resolve this Problem, either resolve the REQUEST_PROMISED_SHORT Problem at the supplying Site, or reduce the receiving_plan Quantity such that it matches the supplying Site's delivery_promise's Quantity.

This is in the SUPPLY and SUPPLY_PROMISE Problem_Sets categories, as well as SUPPLY_PROMISED_SHORT.

The SUPPLY_PROMISED_SHORT model has fields that reference these models :
Operation_Plan, Item_Request, Item_Promise, Item_Acceptance,
Delivery_Request, Delivery_Promise, Delivery_Acceptance.

receiving_plan -- a Operation_Plan field of model
SUPPLY_PROMISED_SHORT

This is the Operation_Plan that will receive the delivery at this Site. Often, this is the Operation_Plan that issued the item_request.

Properties: Export-Only Field

item_request -- a Item_Request field of model SUPPLY_PROMISED_SHORT

The Item_Request whose item_promise has less quantity.

Properties: Export-Only Field

item_promise -- a Item_Promise field of model SUPPLY_PROMISED_SHORT

The Item_Promise whose quantity is less than its item_request.

Properties: Export-Only Field

item_acceptance -- a Item_Acceptance field of model
SUPPLY_PROMISED_SHORT

The Item_Acceptance for the item_request.

Properties: Export-Only Field

Extensions	SUPPLY_PROMISED_EXCESS Extension
------------	----------------------------------

delivery_request - - - *a Delivery_Request field of model SUPPLY_PROMISED_SHORT*

This is simply shorthand for 'item_request.owner'.
Properties: Export-Only Field

delivery_promise - - - *a Delivery_Promise field of model SUPPLY_PROMISED_SHORT*
This is simply shorthand for 'item_promise.owner'.
Properties: Export-Only Field

delivery_acceptance - - - *a Delivery_Acceptance field of model SUPPLY_PROMISED_SHORT*
This is simply shorthand for 'item_acceptance.owner'.
Properties: Export-Only Field

request - - - *a Request field of model SUPPLY_PROMISED_SHORT*
This is simply shorthand for 'item_request.delivery_request.owner'.
Properties: Export-Only Field

promise - - - *a Promise field of model SUPPLY_PROMISED_SHORT*
This is simply shorthand for 'item_promise.delivery_promise.owner'.
Properties: Export-Only Field

acceptance - - - *a Acceptance field of model SUPPLY_PROMISED_SHORT*
This is simply shorthand for 'item_acceptance.delivery_acceptance.owner'.
Properties: Export-Only Field

SUPPLY_PROMISED_EXCESS - - - *a category extension of model Problem*

A SUPPLY_PROMISED_EXCESS Problem indicates that the 'item_promise' has more 'quantity' than its the 'receiving_plan'. The Promise is for more than requested. This Problem generally corresponds to a REQUEST_PROMISED_EXCESS Problem at the supplying Site.

To resolve this Problem, either resolve the REQUEST_PROMISED_EXCESS Problem at the supplying Site, or increase the 'receiving_plan' Quantity such that it matches the supplying Site's 'delivery_promise's Quantity.

This is in the SUPPLY and SUPPLY_PROMISE Problem_Set categories, as well as SUPPLY_PROMISED_EXCESS.

Extensions	SUPPLY_PROMISED_EXCESS Extension
------------	----------------------------------

The SUPPLY_PROMISED_EXCESS model has fields that references these models :
Operation_Plan, Item_Request, Item_Promise, Item_Acceptance,
Delivery_Request, Delivery_Promise, Delivery_Acceptance.

receiving_plan - - - *a Operation_Plan field of model SUPPLY_PROMISED_EXCESS*

This is the Operation_Plan that will receive the delivery at this Site. Often, this is the Operation_Plan that issued the 'item_request'.
Properties: Export-Only Field

item_request - - - *a Item_Request field of model SUPPLY_PROMISED_EXCESS*
The Item_Request whose 'item_promise' has excess 'quantity'.
Properties: Export-Only Field

item_promise - - - *a Item_Promise field of model SUPPLY_PROMISED_EXCESS*
The Item_Promise whose 'quantity' is more than its 'item_request'.
Properties: Export-Only Field

item_acceptance - - - *a Item_Acceptance field of model SUPPLY_PROMISED_EXCESS*
The Item_Acceptance for the 'item_request'.
Properties: Export-Only Field

delivery_request - - - *a Delivery_Request field of model SUPPLY_PROMISED_EXCESS*
This is simply shorthand for 'item_request.owner'.
Properties: Export-Only Field

delivery_promise - - - *a Delivery_Promise field of model SUPPLY_PROMISED_EXCESS*
This is simply shorthand for 'item_promise.owner'.
Properties: Export-Only Field

delivery_acceptance - - - *a Delivery_Acceptance field of model SUPPLY_PROMISED_EXCESS*
This is simply shorthand for 'item_acceptance.owner'.
Properties: Export-Only Field

request - - - *a Request field of model SUPPLY_PROMISED_EXCESS*
This is simply shorthand for 'item_request.delivery_request.owner'.

Extensions	UNIDENTIFIED_OP_STATE Extension
------------	---------------------------------

Properties: Export-Only Field

promise - - - *a Promise field of model SUPPLY_PROMISED_EXCESS*

This is simply shorthand for 'item_promise.delivery_promise.owner'.

Properties: Export-Only Field

acceptance - - - *a Acceptance field of model SUPPLY_PROMISED_EXCESS*

This is simply shorthand for 'item_acceptance.delivery_acceptance.owner'.

Properties: Export-Only Field

UNIDENTIFIED_OP_STATE - - *a category extension of model Problem*

An UNIDENTIFIED_OP_STATE problem indicates that the 'operation_state' has not been able to identify the matching Op_Plan to attached itself to.

To resolve this problem Operation_State should identify an Op_Plan using logic provided by it's identifier extension and attach itself to the selected Op_Plan.

The UNIDENTIFIED_OP_STATE model has fields that references these models :

Operation_State.

operation_state - - - *a Operation_State field of model*

UNIDENTIFIED_OP_STATE

The Operation_State that has this UNIDENTIFIED_OP_STATE problem.

Properties: Export-Only Field

UNCONSOLIDATED - - *a category extension of model Problem*

Unconsolidated Operation_Plans exist in this resource_plan.

The UNCONSOLIDATED model has fields that references these models :

Resource_Plan.

resource_plan - - - *a Resource_Plan field of model UNCONSOLIDATED*

The resource_plan on which this problem is flagged

Properties: Export-Only Field

operation_plans - - - *a List(Operation_Plan) field of model UNCONSOLIDATED*

The list of operation_plans

Properties: Export-Only Field

Extensions	UNCOORDINATED Extension
------------	-------------------------

UNCOORDINATED - - *a category extension of model Problem*

The uncoordination is not coordinated, i.e., all the operation_plans in this consolidation do not start at the same time.

The UNCOORDINATED model has fields that references these models :

Resource_Plan, Consolidation.

resource_plan - - - *a Resource_Plan field of model UNCOORDINATED*

The resource_plan on which this problem is flagged

Properties: Export-Only Field

operation_plans - - - *a List(Operation_Plan) field of model UNCOORDINATED*

The list of operation_plans

Properties: Export-Only Field

consolidation - - - *a Consolidation field of model UNCOORDINATED*

The consolidation that has this problem

Properties: Export-Only Field

CONSOLIDATION_OVERSIZE - - *a category extension of model Problem*

The Operation_Plans in this Consolidation exceeds the max_size of the Consolidation in one of the dimensions of its size

The CONSOLIDATION_OVERSIZE model has fields that references these models :

Resource_Plan, Consolidation.

resource_plan - - - *a Resource_Plan field of model*

CONSOLIDATION_OVERSIZE

The resource_plan on which this problem is flagged

Properties: Export-Only Field

operation_plans - - - *a List(Operation_Plan) field of model*

CONSOLIDATION_OVERSIZE

The list of operation_plans

Properties: Export-Only Field

consolidation - - - *a Consolidation field of model CONSOLIDATION_OVERSIZE*

The consolidation that has this problem

Properties: Export-Only Field

<i>Extensions</i>	<i>CONSOLIDATION_UNDERSIZE Extension</i>
-------------------	--

oversize_dimensions - - *a List(Symbol) field of model*
CONSOLIDATION_OVERSIZE
The dimensions on which oversize occurred
Properties: Export-Only Field

oversize (Symbol) - - *a Percentage field of model* **CONSOLIDATION_OVERSIZE**
The percentage oversize in a dimension 'name'
Properties: Export-Only Field

CONSOLIDATION_UNDERSIZE - - *a category extension of model* Problem

The **Operation_Plan** in this Consolidation consume less than the min_size of the Consolidation in one of the dimensions of its size

The **CONSOLIDATION_UNDERSIZE** model has fields that references these models :
Resource_Plan, Consolidation.

resource_plan - - *a Resource_Plan field of model*
CONSOLIDATION_UNDERSIZE
The resource_plan on which this problem is flagged
Properties: Export-Only Field

operation_plans - - *a List(Operation_Plan) field of model*
CONSOLIDATION_UNDERSIZE
The list of operation_plans
Properties: Export-Only Field

consolidation - - *a Consolidation field of model*
CONSOLIDATION_UNDERSIZE
The consolidation that has this problem
Properties: Export-Only Field

undersize_dimensions - - *a List(Symbol) field of model*
CONSOLIDATION_UNDERSIZE
The dimensions on which undersize occurred
Properties: Export-Only Field

undersize (Symbol) - - *a Percentage field of model*
CONSOLIDATION_UNDERSIZE
The percentage oversize in a dimension 'name'

<i>Extensions</i>	<i>OVERLOAD Extension</i>
-------------------	---------------------------

Properties: Export-Only Field

OVERLOAD - - *a category extension of model* Problem

An **OVERLOAD** Problem indicates that the load planned during these 'dates' on the 'resource_plan' is greater than the capacity during these 'dates'. The 'excess_load_time' gives the severity of the **OVERLOAD** in actual hours, and the 'excess_load_std_time' gives the severity of the **OVERLOAD** in standard hours.

The **OVERLOAD** model has fields that references these models :
Resource_Plan.

resource_plan - - *a Resource_Plan field of model* **OVERLOAD**
The **Resource_Plan** with this **OVERLOAD** Problem.
Properties: Export-Only Field

overload_time - - *a Time field of model* **OVERLOAD**
The excess 'load_time' during the 'dates' of this Problem. This is 'resource_plan.load_time(dates)' - 'resource_plan.available_time(dates)'.
Properties: Export-Only Field

overload_std_time - - *a Time field of model* **OVERLOAD**
The excess 'load_std_time' during the 'dates' of this Problem. This is 'resource_plan.load_std_time(dates)' - 'resource_plan.capacity(dates)'.
Properties: Export-Only Field

OVERSIZE - - *a category extension of model* Problem

An **OVERSIZE** Problem indicates that the load planned during these 'dates' on the 'resource_plan' has size greater than available.

The **OVERSIZE** model has fields that references these models :
Resource_Plan.

resource_plan - - *a Resource_Plan field of model* **OVERSIZE**
The **Resource_Plan** with this **OVERSIZE** Problem.
Properties: Export-Only Field

Extensions	BUCKET_OVERSIZE Extension
------------	---------------------------

oversize -- a Percentage field of model OVERSIZE

The loads on the resource_plan during these dates exceed the size of the Resource by this Percentage. A value of 0% indicates no excess (no Problem). A value of 100% indicates that the peak size usage during dates is twice the available size. Note that a size extension could be considering multiple dimensions of size; this can indicate the maximum oversize of any dimension; or if the dimensions can naturally be combined (volume from 3 lengths), it can indicate the combined figure.

Properties: Export-Only Field

BUCKET_OVERSIZE -- a category extension of model Problem

This problem is defined for SHARED_USE Load_Policy only.

A BUCKET_OVERSIZE Problem indicates that the load planned during a time bucket on the resource_plan is greater than max_bucket_load, specified in the SHARED_USE load_policy of the resource. The problem period is the same as the bucket period.

The BUCKET_OVERSIZE model has fields that references these models :
Resource_Plan.

resource_plan -- a Resource_Plan field of model BUCKET_OVERSIZE
The Resource_Plan with this BUCKET_OVERSIZE Problem.
Properties: Export-Only Field

bucket_oversize -- a Percentage field of model BUCKET_OVERSIZE
(load_size/available_capacity - max_bucket_load) during the problem period.
available_capacity is the size of the resource in the problem period. load_size is the sum of contribution of all the load_plans in this period

Properties: Export-Only Field

UNDERLOAD -- a category extension of model Problem

This problem is defined for SHARED_USE Load_Policy only.

An UNDERLOAD Problem indicates that the load planned during the problem period on the resource_plan is less than desired. The minimum desired load on this resource is specified as min_load in the SHARED_USE Load_Policy. The problem period begins at Plan_start and ends at min_load_fence specified in the SHARED_USE Load_Policy.

Extensions	NEGATIVE_ON_HAND Extension
------------	----------------------------

Be careful with this Problem. Utilizing Resources just to utilize them is usually bad practice. However, there are times when it is known that a Resource must stay utilized (it is a fluctuating bottleneck), and it is useful to work ahead to handle unexpected variation.

The UNDERLOAD model has fields that references these models :
Resource_Plan.

resource_plan -- a Resource_Plan field of model UNDERLOAD
The Resource_Plan with this UNDERLOAD Problem.

Properties: Export-Only Field

underload -- a Percentage field of model UNDERLOAD
(min_load - (load_size/available_capacity)) during the problem period. This is the percentage of the resource capacity that is not utilized. available_capacity is calculated from the size of the resource in the problem period. load_size is the sum of contribution of all the load_plans in this period

Properties: Export-Only Field

NEGATIVE_ON_HAND -- a category extension of model Problem

A NEGATIVE_ON_HAND Problem indicates that the flow planned during these dates on the buffer_plan results in the on_hand being less than zero. Such a Problem is inherently infeasible.

The NEGATIVE_ON_HAND model has fields that references these models :
Buffer_Plan.

buffer_plan -- a Buffer_Plan field of model NEGATIVE_ON_HAND
The Buffer_Plan with this NEGATIVE_ON_HAND Problem.
Properties: Export-Only Field

low_on_hand -- a Quantity field of model NEGATIVE_ON_HAND
The lowest (most negative) value of buffer_plan_on_hand during these dates. This Quantity is converted to the unit of the Buffer.
Properties: Export-Only Field

deficit -- a Quantity field of model NEGATIVE_ON_HAND
The additional Quantity needed during these dates to reach an acceptable on_hand level. This Quantity is converted to the unit of the Buffer.
Properties: Export-Only Field

<i>Extensions</i>	<i>OVER_FLOW_LIMIT Extension</i>
-------------------	----------------------------------

OVER_FLOW_LIMIT -- a category extension of model Problem

A OVER_FLOW_LIMIT Problem indicates that the flow planned during these 'dates' on the *buffer_plan* results in the *on_hand* being less than zero. These problems are identical to NEGATIVE_ON_HAND problems except that these problems are generated by the FLOW_LIMIT_CALENDAR flow policy. This flow policy is to be used in modelling resources (unit capacity buffers).

The OVER_FLOW_LIMIT model has fields that references these models :
Buffer_Plan.

buffer_plan -- a *Buffer_Plan* field of model OVER_FLOW_LIMIT
 The *Buffer_Plan* with this OVER_FLOW_LIMIT Problem.

Properties: Export-Only Field

low_on_hand -- a *Quantity* field of model OVER_FLOW_LIMIT

The lowest (most negative) value of *buffer_plan.on_hand* during these dates. This Quantity is converted to the unit of the Buffer.

Properties: Export-Only Field

deficit -- a *Quantity* field of model OVER_FLOW_LIMIT

The additional Quantity needed during these dates to reach an acceptable *on_hand* level. This Quantity is converted to the unit of the Buffer.

Properties: Export-Only Field

NEGATIVE_ON_HAND_AT_END -- a category extension of model Problem

A NEGATIVE_ON_HAND_AT_END Problem indicates that the flow planned on the *buffer_plan* results in the *on_hand* being less than zero after the end of the planning horizon. This special case of NEGATIVE_ON_HAND is often used to propagate demand upstream. Such a Problem is inherently infeasible.

The NEGATIVE_ON_HAND_AT_END model has fields that references these models :
Buffer_Plan.

buffer_plan -- a *Buffer_Plan* field of model NEGATIVE_ON_HAND_AT_END
 The *Buffer_Plan* with this NEGATIVE_ON_HAND Problem.

Properties: Export-Only Field

<i>Extensions</i>	<i>LOT_OVER_CONSUMED Extension</i>
-------------------	------------------------------------

low_on_hand -- a *Quantity* field of model NEGATIVE_ON_HAND_AT_END
 The lowest (most negative) value of *buffer_plan.on_hand* during these dates. This Quantity is converted to the unit of the Buffer.

Properties: Export-Only Field

deficit -- a *Quantity* field of model NEGATIVE_ON_HAND_AT_END

The additional Quantity needed during these dates to reach an acceptable *on_hand* level. This Quantity is converted to the unit of the Buffer.

Properties: Export-Only Field

LOT_OVER_CONSUMED -- a category extension of model Problem

A LOT_OVER_CONSUMED Problem indicates that supplying flows planned for a specific Lot are insufficient for the consuming flows of that Lot. The supplying flows may be smaller than the consuming flows, or they may be later. This is similar to a NEGATIVE_ON_HAND Problem, but is used instead in Lot-for-Lot Buffers to indicate the specific Lot assignment that has the problem.

The LOT_OVER_CONSUMED model has fields that references these models :
Buffer_Plan.

buffer_plan -- a *Buffer_Plan* field of model LOT_OVER_CONSUMED

The *Buffer_Plan* with this LOT_OVER_CONSUMED Problem.

Properties: Export-Only Field

shortage_quantity -- a *Quantity* field of model LOT_OVER_CONSUMED

The quantity difference between the supplying Lots and the consuming Lot. For example, if a 70 unit supplier is paired with a 100 unit consumer, this value will be -30 units. Note that this value could be zero (or even positive) if shortage_time is not zero.

Properties: Export-Only Field

shortage_time -- a *Time* field of model LOT_OVER_CONSUMED

If a Lot is planned earlier than the supplying Lots it consumes from, this is the amount of time between the planned start date of the consumer and the planned completion date of the earliest Lot it consumes from. If the consuming Lot is planned as late as, or later than, all the Lots it consumes from, this field will be zero. This could happen if shortage_quantity is less than zero.

Properties: Export-Only Field

LOT_NOT_CONSUMED -- a category extension of model Problem

A LOT_NOT_CONSUMED Problem indicates that supplying flows planned for a specific Lot are not allocated to any consuming flows.

The LOT_NOT_CONSUMED model has fields that references these models :
Buffer_Plan.

buffer_plan - - a Buffer_Plan field of model LOT_NOT_CONSUMED
The Buffer_Plan with this LOT_NOT_CONSUMED Problem.
Properties: Export-Only Field

LOT_NOT_PRODUCED - - a category extension of model Problem

A LOT_NOT_PRODUCED Problem indicates that consuming flows planned for a Buffer are not allocated to any producing flows.

The LOT_NOT_PRODUCED model has fields that references these models :
Buffer_Plan.

buffer_plan - - a Buffer_Plan field of model LOT_NOT_PRODUCED
The Buffer_Plan with this LOT_NOT_CONSUMED Problem.
Properties: Export-Only Field

LOT_OVER_PRODUCED - - a category extension of model Problem

A LOT_OVER_PRODUCED Problem indicates that supplying flows planned for a specific Lot are excessive for the consuming flows of that Lot. The supplying flows may be larger than the consuming flows, or they may be earlier. This is similar to an EXCESS_ON_HAND Problem, but is used instead in Lot-for-Lot Buffers to indicate the specific Lot assignment that has the problem.

The LOT_OVER_PRODUCED model has fields that references these models :
Buffer_Plan.

buffer_plan - - a Buffer_Plan field of model LOT_OVER_PRODUCED
The Buffer_Plan with this LOW_ON_HAND Problem.
Properties: Export-Only Field

excess_quantity - - a Quantity field of model LOT_OVER_PRODUCED
The quantity difference between the producing Lots and the consuming Lot. For example, if a 70 unit producer is paired with a 50 unit consumer, this value will be 20 units. Note that this value could be zero (or even negative) if excess_time is not zero.
Properties: Export-Only Field

excess_time - - a Time field of model LOT_OVER_PRODUCED

If a Lot is planned later than the supplying Lots it consumes from, this is the amount of time between the planned start date of the consumer and the planned completion date of the earliest Lot it consumes from. If the consuming Lot is planned as early as, or earlier than, all the Lots it consumes from, this field will be zero. This could happen if 'excess_quantity' is greater than zero.

Properties: Export-Only Field

LOW_ON_HAND - - a category extension of model Problem

A LOW_ON_HAND Problem indicates that the flow planned during these 'dates' on the buffer_plan results in the 'on_hand' being less than 'min_on_hand' or other on_hand limits. Such Problems are typically feasible, but undesirable for various reasons (mostly safety stock and service level related reasons).

The LOW_ON_HAND model has fields that references these models :
Buffer_Plan.

buffer_plan - - a Buffer_Plan field of model LOW_ON_HAND
The Buffer_Plan with this LOW_ON_HAND Problem.
Properties: Export-Only Field

low_on_hand - - a Quantity field of model LOW_ON_HAND

The lowest value of buffer_plan.on_hand during these 'dates'. This Quantity is converted to the unit of the Buffer.
Properties: Export-Only Field

deficit - - a Quantity field of model LOW_ON_HAND

The additional Quantity needed during these 'dates' to reach an acceptable on_hand level. This Quantity is converted to the unit of the Buffer.
Properties: Export-Only Field

EXCESS_ON_HAND - - a category extension of model Problem

An EXCESS_ON_HAND Problem indicates that the flow planned during these 'dates' on the buffer_plan results in the 'on_hand' being in excess of desired levels. Such a Problem is typically feasible, but undesirable for various reasons (to prevent excess inventory and all the related costs).

The EXCESS_ON_HAND model has fields that references these models :
Buffer_Plan.

<i>Extensions</i>	<i>EXCESS_ON_HAND_AT_END Extension</i>
-------------------	--

buffer_plan -- *a Buffer_Plan field of model EXCESS_ON_HAND*
The Buffer_Plan with this EXCESS_ON_HAND Problem.

Properties: Export-Only Field

excess -- *a Quantity field of model EXCESS_ON_HAND*
The largest excess Quantity that is in buffer_plan.on_hand during these dates. Note that this is the Quantity beyond the desired levels and beyond the permissible excess levels. This Quantity is converted to the unit of the Buffer.

Properties: Export-Only Field

EXCESS_ON_HAND_AT_END -- *a category extension of model Problem*

An EXCESS_ON_HAND_AT_END Problem indicates that the flow planned on the buffer_plan results in the on_hand being in excess of desired levels at the end of the planning horizon. Such a Problem is typically feasible, but undesirable for various reasons (to prevent excess inventory and all the related costs).

The EXCESS_ON_HAND_AT_END model has fields that reference these models:
Buffer_Plan.

buffer_plan -- *a Buffer_Plan field of model EXCESS_ON_HAND_AT_END*
The Buffer_Plan with this EXCESS_ON_HAND_AT_END Problem.

Properties: Export-Only Field

excess -- *a Quantity field of model EXCESS_ON_HAND_AT_END*
The excess Quantity that is in buffer_plan at the end of the planning horizon. Note that this is the Quantity beyond the desired levels and beyond the permissible excess levels. This Quantity is converted to the unit of the Buffer.

Properties: Export-Only Field

<i>Extensions</i>	<i>Active_Strategy Extensions</i>
-------------------	-----------------------------------

10.23 Active_Strategy Extensions

10.23.1 termination extensions of model Active_Strategy

NO_PROBLEMS -- *a termination extension of model Active_Strategy*

A NO_PROBLEMS termination Active_Strategy will run until it finds no improvements to the plan for more than max_stable_time or until it has run in all for more than max_run_time. It specifies no additional termination condition.

TARGET_ACHIEVED -- *a termination extension of model Active_Strategy*

A TARGET_ACHIEVED termination Active_Strategy will run until it achieves the target as specified by the goal extension fields. Note that some goal extensions specify no target, and thus will never be terminated by achieving the target.

RESOLVE_COUNT_EXCEEDED -- *a termination extension of model Active_Strategy*

A RESOLVE_COUNT_EXCEEDED termination Strategy will run until it has attempted to resolve max_resolve_count problems.

MANUAL -- *a termination extension of model Active_Strategy*

A MANUAL termination Strategy will run until user hits the 'done' button (sets 'done' to 'true'). This is generally used as a sub_strategy to pause for the user to make some decisions. As an Active_Strategy, it can present to the user the relevant Problems and Goal measures. Pressing 'done' ends that sub_strategy, and continues on with the next.

done -- *a Logical field of model MANUAL*

This Active_Strategy will terminate if 'done' is set 'true'.

Default: false

10.23.2 execution extensions of model Active_Strategy

SEQUENCE_RUN_ONCE -- *a execution extension of model Active_Strategy*

A SEQUENCE_RUN_ONCE strategy executes its sub_strategies in a particular order. Each Ordered_Sub_Strategy will be run exactly once till it terminated. The order is defined using 'sequence' field of Ordered_Sub_Strategy model. The lower the number the earlier the strategy is performed.

Extensions	SEQUENCE_RUN_MULTIPLE Extension
------------	---------------------------------

sub_strategies - - - *a List(Active_Strategy) field of model SEQUENCE_RUN_ONCE*
 the list of Active_Strategy that will be performed sequentially
 Properties: Export-Only Field

current_sub - - - *a Number field of model SEQUENCE_RUN_ONCE*
 current sub_strategy that is running
 Properties: Export-Only Field

SEQUENCE_RUN_MULTIPLE - - - *a execution extension of model Active_Strategy*

A SEQUENCE_RUN_MULTIPLE strategy executes its sub_strategies in a particular order. Each Ordered_Sub_Strategy will be run exactly once till it terminated. The order is defined using 'sequence' field of Ordered_Sub_Strategy model. The lower the number the earlier the strategy is performed.

sub_strategies - - - *a List(Active_Strategy) field of model SEQUENCE_RUN_MULTIPLE*
 the list of Active_Strategy that will be performed sequentially
 Properties: Export-Only Field

current_sub - - - *a Number field of model SEQUENCE_RUN_MULTIPLE*
 current sub_strategy that is running
 Properties: Export-Only Field

BEFORE_AND_AFTER - - - *a execution extension of model Active_Strategy*

See extension BEFORE_AND_AFTER for Strategy

The BEFORE_AND_AFTER model has fields that references these models :
 Active_Strategy.

before_run - - - *a Active_Strategy field of model BEFORE_AND_AFTER*
 This Strategy will run before the start of 'owner' Strategy run.
 Default: [unspecified]

Properties: Export-Only Field

before_search - - - *a Active_Strategy field of model BEFORE_AND_AFTER*
 This strategy will run before the start of each search.
 Default: [unspecified]

Properties: Export-Only Field

Extensions	SEQUENTIAL_ALTERNATES Extension
------------	---------------------------------

after_resolve - - - *a Active_Strategy field of model BEFORE_AND_AFTER*
 This strategy will run after each problem is resolved.
 Default: [unspecified]
 Properties: Export-Only Field

after_search - - - *a Active_Strategy field of model BEFORE_AND_AFTER*
 This strategy will run after each search is complete.
 Default: [unspecified]
 Properties: Export-Only Field

after_success - - - *a Active_Strategy field of model BEFORE_AND_AFTER*
 This Strategy will run after each successful search. It will only run if a search results in a better Plan, where better is defined by the Strategy 'goals', then the search is termed a "success".
 Default: [unspecified]

Properties: Export-Only Field

after_run - - - *a Active_Strategy field of model BEFORE_AND_AFTER*
 This Strategy will run at the completion of 'owner' Strategy run.
 Default: [unspecified]

Properties: Export-Only Field

SEQUENTIAL_ALTERNATES - - - *a execution extension of model Active_Strategy*

See extension SEQUENTIAL_ALTERNATES for Strategy

The SEQUENTIAL_ALTERNATES model has fields that references these models :
 Active_Strategy.

propagation - - - *a Active_Strategy field of model SEQUENTIAL_ALTERNATES*
 This Strategy will run after each resolve of the parent.
 Default: [unspecified]

Properties: Export-Only Field

evaluation - - - *a Active_Strategy field of model SEQUENTIAL_ALTERNATES*
 The goal function of this strategy is used to evaluate each alternate.
 Default: [unspecified]

Properties: Export-Only Field

cleanup - - - *a Active_Strategy field of model SEQUENTIAL_ALTERNATES*
 This strategy will run after each failed alternate, before moving to the next alternate.

Extensions	SEQUENTIAL_ALTERNATES_KEEP_BEST Extension
------------	---

Default: [unspecified]
Properties: Export-Only Field

min_alt -- *a Integer field of model SEQUENTIAL_ALTERNATES*
The min_alt and max_alt defines the range of alternate operations considered. For example, if min_alt = max_alt = 0, only the first alternate is allowed. The min and max are duplicated in Active_Execution because of the possibility of recursive strategy, where each subsequent invocation may have different values.
Default: 0

max_alt -- *a Integer field of model SEQUENTIAL_ALTERNATES*
See notes in min_alt.
Default: 100

current_alt -- *a Integer field of model SEQUENTIAL_ALTERNATES*
The current alternate allowed
Properties: Export-Only Field

SEQUENTIAL_ALTERNATES_KEEP_BEST -- *a execution extension of model Active_Strategy*

See extension SEQUENTIAL_ALTERNATES_KEEP_BEST for Strategy

The SEQUENTIAL_ALTERNATES_KEEP_BEST model has fields that references these models :
Active_Strategy.

propagation -- *a Active_Strategy field of model SEQUENTIAL_ALTERNATES_KEEP_BEST*
This Strategy will run after each resolve of the parent.
Default: [unspecified]
Properties: Export-Only Field

evaluation -- *a Active_Strategy field of model SEQUENTIAL_ALTERNATES_KEEP_BEST*
The goal function of this strategy is used to evaluate each alternate.
Default: [unspecified]
Properties: Export-Only Field

Extensions	PROPORTIONAL_RESOLVES Extension
------------	---------------------------------

min_alt -- *a Integer field of model SEQUENTIAL_ALTERNATES_KEEP_BEST*
The min_alt and max_alt defines the range of alternate operations considered. For example, if min_alt = max_alt = 0, only the first alternate is allowed. The min and max are duplicated in Active_Execution because of the possibility of recursive strategy, where each subsequent invocation may have different values.
Default: 0

max_alt -- *a Integer field of model SEQUENTIAL_ALTERNATES_KEEP_BEST*
See notes in min_alt.
Default: 100

PROPORTIONAL_RESOLVES -- *a execution extension of model Active_Strategy*

See extension PROPORTIONAL_RESOLVES for Strategy

sub_strategies -- *a List(Active_Strategy) field of model PROPORTIONAL_RESOLVES*
The list of Active_Strategy that will be performed sequentially
Properties: Export-Only Field

ORDERED_RESOLVES -- *a execution extension of model Active_Strategy*

A Ordered_Resolve strategy executes the highest rank Ranked_Sub_Strategy with a problem after each resolve.

sub_strategies -- *a List(Active_Strategy) field of model ORDERED_RESOLVES*
A List of Active_Strategy that will be executed during the run of owner
Active_Strategy
Properties: Export-Only Field

10.23.3 problem_selection extensions of model Active_Strategy

PROPORTIONAL_INTERACTION -- *a problem_selection extension of model Active_Strategy*

The PROPORTIONAL_INTERACTION problem_selection extension selects the problem to solve using following logic.

1. It first selects the Problem_Set probabilistically, weighted by Problem_Set.focus times the total interaction of problems within that Problem_Set. 2. Choose a problem within selected Problem_Set probabilistically, weighted by the interaction value of Problem. Otherwise return the current chosen problem.

Extensions	EARLIEST_PROBLEM_START Extension
------------	----------------------------------

At Strategy.heal == 0, it will first select the Problem_Set with the largest Problem_Set.focus value and within that it will select the Problem with the largest interaction value.

When extension's behavior becomes deterministic, it will break the ties by selecting the problem with the earliest startdate. If there are multiple problems with the same start date, it will break the further ties by selecting problem randomly.

EARLIEST_PROBLEM_START -- a problem_selection extension of model Active_Strategy

The EARLIEST_PROBLEM_START problem_selection extension selects the problem probabilistically, weighted by problem's start date. As the Strategy.heal approaches zero, probability of selecting the problem with the earliest start date increases continuously. At Strategy.selection_heal == 0, it will be completely deterministic and will always select the problem with the earliest start date first.

When the selection becomes deterministic, it will break the tie among the same start date problems, by selecting the problem with the highest interaction. If there are multiple problems with the same interaction value, it will break the further tie by selecting a problem randomly.

SORT_BY_EXPRESSION -- a problem_selection extension of model Active_Strategy

The SORT_BY_EXPRESSION problem_selection extension selects the first problem from the sorted problem list. The problem list is sorted using the user specified sort_criteria expression.

Example Usage:

sort_criteria

Extensions	Active_Goal Extensions
------------	------------------------

10.24 Active_Goal Extensions

10.24.1 goal extensions of model Active_Goal

FEASIBILITY -- a goal extension of model Active_Goal

The goal is to minimize the sum of 'interaction' of infeasible Problems, ultimately driving it to zero by eliminating all infeasible Problems.

total_interaction -- a Number field of model FEASIBILITY

The current level for the sum of 'interaction' of all infeasible Problems. If this value is less than or equal to strategy_goal.target_interaction, then the target has been met (though the goal to minimize it to zero remains).

This is typically used in conjunction with target-oriented termination extensions.

Properties: Export-Only Field

MINIMIZE_PROBLEM_COUNT -- a goal extension of model Active_Goal

The goal is to minimize the number of problems ultimately driving it to zero by eliminating all Problems.

problem_count -- a Number field of model MINIMIZE_PROBLEM_COUNT

The current number of problems. If this value is less than or equal to strategy_goal.problem_count, then the target has been met (though the goal to minimize it to zero remains).

This is typically used in conjunction with target-oriented termination extensions.

Properties: Export-Only Field

MINIMIZE_PROBLEMS -- a goal extension of model Active_Goal

The goal is to minimize the sum of the weights of all problems, ultimately driving it to zero by eliminating all Problems.

problems -- a Number field of model MINIMIZE_PROBLEMS

The sum of the weights of the current problems. If this value is less than or equal to strategy_goal.problems, then the target has been met (though the goal to minimize it to zero remains).

This is typically used in conjunction with target-oriented termination extensions.

Default: 0

<i>Extensions</i>	<i>MINIMIZE_LATENCY Extension</i>
-------------------	-----------------------------------

Properties: Export-Only Field

MINIMIZE_LATENCY -- a goal extension of model Active_Goal

The goal is to minimize the sum of 'latency' of all requests.

total_latency - - a Number field of model MINIMIZE_LATENCY
The current level for the sum of 'latency' of all requests. If this value is less than or equal to 'strategy_goal_latency', then the target has been met (though the goal to minimize it to zero remains).

This is typically used in conjunction with target-oriented termination extensions.
Properties: Export-Only Field

WEIGHTED_LATENCY -- a goal extension of model Active_Goal

The goal is to minimize the weighted sum of 'latency' of all requests.

total_latency - - a Number field of model WEIGHTED_LATENCY
The current level for the sum of 'latency' of all requests. If this value is less than or equal to 'strategy_goal_latency', then the target has been met (though the goal to minimize it to zero remains).

This is typically used in conjunction with target-oriented termination extensions.
Properties: Export-Only Field

WEIGHTED_SHORTNESS -- a goal extension of model Active_Goal

The goal is to minimize the weighted sum of 'latency' of all requests.

total_shortness - - a Number field of model WEIGHTED_SHORTNESS
The current level for the sum of 'shortness' of all requests. If this value is less than or equal to 'strategy_goal_target_shortness', then the target has been met (though the goal to minimize it to zero remains).

This is typically used in conjunction with target-oriented termination extensions.
Properties: Export-Only Field

MINIMIZE_SHORTNESS -- a goal extension of model Active_Goal

The goal is to minimize the sum of 'shortness' of all requests

<i>Extensions</i>	<i>MINIMIZE_COST Extension</i>
-------------------	--------------------------------

total_shortness - - a Number field of model MINIMIZE_SHORTNESS
The current level for the sum of 'shortness' of all requests. If this value is less than or equal to 'strategy_goal_target_shortness', then the target has been met (though the goal to minimize it to zero remains).

This is typically used in conjunction with target-oriented termination extensions.
Properties: Export-Only Field

MINIMIZE_COST -- a goal extension of model Active_Goal

The goal is to minimize the cost of the current plan

total_cost - - a Number field of model MINIMIZE_COST
The current level for the 'cost' of the plan. If this value is less than or equal to 'strategy_goal_target_cost', then the target has been met (though the goal to minimize it to zero remains).

This is typically used in conjunction with target-oriented termination extensions.
Default: 0
Properties: Export-Only Field

MAXIMIZE_PROFIT -- a goal extension of model Active_Goal

The goal is to maximize 'profit' (the difference between 'revenue' and 'cost').

total_profit - - a Number field of model MAXIMIZE_PROFIT
The current level for 'profit' (the difference between 'revenue' and 'cost'). If this value is greater than or equal to 'strategy_goal_target_profit', then the target has been met (though the goal to maximize it remains).

This is typically used in conjunction with target-oriented termination extensions.
Default: 0
Properties: Export-Only Field

MAXIMIZE_REVENUE -- a goal extension of model Active_Goal

The goal is to maximize 'revenue'.

total_revenue - - a Number field of model MAXIMIZE_REVENUE
The current level for 'revenue'. If this value is greater than or equal to 'strategy_goal_target_revenue', then the target has been met (though the goal to maximize it remains).

Extensions	MAXIMIZE REVENUE Extension
------------	----------------------------

This is typically used in conjunction with target-oriented termination extensions.
Default: 0
Properties: Export-Only Field

Extensions	Item Extensions
------------	-----------------

10.25 Item Extensions

10.25.1 spec extensions of model Item

STANDARD -- a spec extension of model Item

All Configurations of this Item are interchangeable -- fully equivalent for planning purposes. No additional fields are needed in order to specify the Lot to find or build.

CUSTOM -- a spec extension of model Item

The Item is specified as an order-specific item, such as is the norm in engineer-to-order businesses. In these businesses, the Items are designed for each customer. However, modeling a new Item for each customer order would be pointless. Thus, general families of items are modeled with Items, and the order-specific Items are simply the Configurations produced. Thus, no two Configurations of this Item will be interchangeable.

Note that no Lot-specific information is maintained. However, 'lots_tracked' will be "true" because they are each assumed distinct.

<i>Extensions</i>	<i>Skill Extensions</i>
-------------------	-------------------------

10.26 Skill Extensions

10.26.1 selection extensions of model Skill

PRIMARY -- *a selection extension of model Skill*

The 'primary' Resource is selected by default. The others can be chosen manually.

The PRIMARY model has fields that references these models :
Resource.

primary -- *a Resource field of model PRIMARY*

The primary Resource that is selected by default.

Default: [unspecified]

PREFER_PRIMARY -- *a selection extension of model Skill*

The 'primary' Resource is selected by default. The others could be chosen while planning, when moving to an alternate resource. These other alternate resources are chosen at random. While we chose an alternate resource, if the load is already on an alternate resource (a resource other than the primary), then this always chooses the primary. The alternate resources can also be chosen manually.

The PREFER_PRIMARY model has fields that references these models :
Resource.

primary -- *a Resource field of model PREFER_PRIMARY*

The primary Resource that is selected by default.

Default: [unspecified]

EVEN -- *a selection extension of model Skill*

The 'resources' are selected with even probability. Note that this is the default for Skills generated for each Resource Family.

MAX_EFFICIENCY -- *a selection extension of model Skill*

The 'resources' are selected in order of maximum 'efficiency'.

<i>Extensions</i>	<i>Skill_Resource Extensions</i>
-------------------	----------------------------------

10.27 Skill_Resource Extensions

10.27.1 efficiency extensions of model Skill_Resource

FIXED -- *a efficiency extension of model Skill_Resource*

A FIXED efficiency Skill_Resource has a fixed efficiency level at all times. The default value is the very common case of 100%.

fixed_efficiency -- *a Percentage field of model FIXED*
The efficiency level of the resource in performing this skill

Default: 100%

CALENDAR -- *a efficiency extension of model Skill_Resource*

A CALENDAR efficiency Skill_Resource has efficiency that will vary over time as specified in a Calendar.

Further, the options for increasing efficiency by incurring additional cost can be modeled. Calendar_Entry's with a non-zero 'charge' are treated as options for increasing efficiency when needed. Uses the Default_Efficiency_Calendar, if efficiency_calendar is not set.

The CALENDAR model has fields that references these models :
Calendar.

efficiency_calendar -- *a Calendar field of model CALENDAR*

A Calendar with PERCENTAGE 'entries', where the Percentage is the efficiency level at those dates. Entries with a non-zero 'charge' are treated as options for increasing efficiency when needed.

Default: [unspecified]

10.28 Configuration Extensions

10.28.1 spec extensions of model Configuration

STANDARD -- a spec extension of model Configuration

All Configurations of this Item are interchangeable -- fully equivalent for planning purposes. No additional fields are needed in order to specify the Lot to find or build.

CUSTOM -- a spec extension of model Configuration

The Item is specified as an order-specific item, such as is the norm in engineer-to-order businesses. In these businesses, the Items are designed for each customer. However, modeling a new Item for each customer order would be pointless. Thus, general families of Items are modeled with Items, and the order-specific Items are simply the Configurations produced. Thus, no two Configurations of this Item will be interchangeable.

Note that no Lot-specific information is maintained. However, 'lots_tracked' will be "true" because they are each assumed distinct.

10.29 Operation_State Extensions

10.29.1 identifier extensions of model Operation_State

EARLIEST -- a identifier extension of model Operation_State

Identifies the earliest non-released Op_Plan for the 'operation' and attaches the State information to it.

The EARLIEST model has fields that references these models :
Operation.

release_name -- a Symbol field of model EARLIEST
The release name of the identified Op_Plan.

Default: [unspecified]

operation -- a Operation field of model EARLIEST
Operation for which the state information is reported.
Default: [unspecified]

top_operation -- a Operation field of model EARLIEST
top most operation in the 'operation' tree.
Default: [unspecified]

10.29.2 state_spec extensions of model Operation_State

STARTED -- a state_spec extension of model Operation_State

A STARTED Operation_State's 'quantity' specifies the cumulative Quantity that has been started by the identified Operation_Plan as of 'date'.

Note that this is a cumulative Quantity, it does not shrink as material is completed. The quantity being processed is completed + scrapped - started. This Operation_State also implies the ARRIVED Quantity is at least as large.

The STARTED model has fields that references these models :
Item.

item -- a Item field of model STARTED
The Item for which the starting 'quantity' is reported.
Default: [unspecified]

Extensions	COMPLETED Extension
------------	---------------------

quantity -- *a Quantity field of model STARTED*
The Quantity that has started execution (production, transportation, etc.), as of 'date'.
If the Item is unspecified, then this Quantity needs to be unitless and it represents the units of operation plan.
Default: 0

COMPLETED -- a state_spec extension of model Operation_State

A COMPLETED Operation_State's 'quantity' specifies the cumulative Quantity that has been completed by the identified Operation_Plan as of 'date'.

Note that this is a cumulative Quantity -- the total ever completed by this Operation. This Operation_State also implies that the ARRIVED and STARTED Quantity's are at least as large as the sum of the SCRAPPED and COMPLETED Quantity's.

The COMPLETED model has fields that references these models :
Item.

Item -- *a Item field of model COMPLETED*
The Item for which the completed quantity is reported.
Default: [unspecified]

quantity -- *a Quantity field of model COMPLETED*
The Quantity that has completed execution (production, transportation, etc.), as of 'date'.

If the Item is unspecified, then this Quantity needs to be unitless and it represents the units of operation plan.
Default: 0

IN_FRONT -- a state_spec extension of model Operation_State

An IN_FRONT Operation_State's 'quantity' specifies the Quantity of this Operation that is "in front" of the Operation, ready to start. It is the Quantity that has arrived but not yet started'.

Note that IN_FRONT state information can only be meaningful if the full snapshot of the super_operation_plan is given. Therefore, IN_FRONT Operation_State's are ignored unless their 'date' is equal to the latest of all Operation_States of all Operation_Plans under the 'top_operation_plan'.

Extensions	IN_FRONT Extension
------------	--------------------

quantity -- *a Quantity field of model IN_FRONT*
The Quantity of this Operation that is standing "in front" of this Operation -- the Quantity that has arrived' but not started'.
Default: 0

<i>Extensions</i>	<i>Request Extensions</i>
-------------------	---------------------------

10.30 Request Extensions

10.30.1 delivery_policy extensions of model Request

FIXED -- a delivery_policy extension of model Request

Fixed delivery policy

10.30.2 delivery_naming extensions of model Request

NUMBERED -- a delivery_naming extension of model Request

Delivery_Requests are named by successive numbers.

<i>Extensions</i>	<i>Delivery_Request Extensions</i>
-------------------	------------------------------------

10.31 Delivery_Request Extensions

10.31.1 promising_policy extensions of model Delivery_Request

BUCKETED_ALL -- a promising_policy extension of model Delivery_Request

This is similar to ALL promising policy, but in addition to that it takes care of consume_earlier, Generic_Products, Alternate_Products, and the Generic_Products that are shared between the forecasts of the items of a delivery request. It is similar to Bucketed_All_Min_Price except that this policy would give quotes from all the matching forecasts.

BUCKETED_ASAP -- a promising_policy extension of model Delivery_Request

This is similar to ASAP promising policy, but in addition to that it takes care of consume_earlier, Generic_Products, Alternate_Products, and the Generic_Products that are shared between the forecasts of the items of a delivery request. It is similar to Bucketed_All_Min_Price except that this policy would give quotes from all the matching forecasts.

ON_TIME -- a promising_policy extension of model Delivery_Request

If an ON_TIME promise_policy Delivery_Request cannot be satisfied, then the promise should only be made for the amount/price that can be delivered on the date requested (it will be promised short). The Delivery_Request should not be promised late.

ALL -- a promising_policy extension of model Delivery_Request

If an ALL promising_policy Delivery_Request cannot be satisfied, then the promise should only be made for the entire amount/price (it will be promised late). The Delivery_Request should not be promised short. This policy will not work for cases of Product consume_earlier set to anything other than the default value.

ALL_ON_TIME -- a promising_policy extension of model Delivery_Request

If an ALL_ON_TIME promising_policy Delivery_Request cannot be satisfied for the entire amount/price on the date requested, then it should not be promised. The Delivery_Request should not be promised short or late. This policy will not work for cases of Product consume_earlier set to anything other than the default value.

ASAP -- a promising_policy extension of model Delivery_Request

If an ASAP promising_policy Delivery_Request cannot be satisfied, then it should be promised so as to deliver as much of the request as possible as soon as possible. This may split the request into a number of deliveries. This policy will not work for cases of Product.consume_earlier set to anything other than the default value.

ASAP_MONTHLY -- a promising_policy extension of model Delivery_Request

If an ASAP_MONTHLY promising_policy Delivery_Request cannot be satisfied, then it should be promised so as to deliver as much of the request as possible on the date requested and the rest will split into monthly deliveries. The monthly deliveries will be as late in the month as required to obtain the maximum amount of the available to promise for that month, but no later. This policy will not work for cases of Product.consume_earlier set to anything other than the default value.

BUCKETED_ALLOCATION -- a promising_policy extension of model Delivery_Request

If a BUCKETED_ALLOCATION promising_policy Delivery_Request cannot be satisfied on the date requested, then it should be promised as to deliver as much of the request as possible on the date requested. The remaining promises will split into bucketed deliveries where the buckets correspond to the forecast horizon. The EARLY bucketed deliveries will each be as late in the bucket as required to obtain the maximum amount of product available in that bucket. For promises in LATE buckets (as well as late promises in the requested bucket), the deliveries will be according to ASAP. That is, there will be one delivery for each date where there is additional quantity available for at least one item request. Preference is given to ON TIME promises first, EARLY promises next, and finally LATE promises. NOTE: Assumes that Product.consume_earlier settings do not span multiple forecast horizon buckets.

BUCKETED_ALL_MIN_PRICE -- a promising_policy extension of model Delivery_Request

This policy's behavior would be similar to that of ALL with some exceptions. One of them currently is that the 'promising_policy_aip' would quote a single product only for a given item_request. If it finds that all the item requests could be satisfied by a date, then that date would be the delivery date of this policy's delivery_aip.

For a given item request, the product that gives all the quantities by the delivery date would be considered. If more than one product could satisfy by the delivery date, then one of them with the minimum 'price' would be chosen.

This policy gives only one delivery_aip. This helps to know the 'delivery_date' of all the quantity's of all the item requests that are being delivered together. Furthermore, it would contain one item_aip for each item_request. Each item_aip would contain multiple product_aip's. The product_aip's 'dates' would be different from each other depending on the aip_entries' dates. But when the consumption takes place using this policy, then multiple delivery_aip's and item_aip's would be created due to and according to the different dates of the product_aip's.

This policy works well with generic products. The incremental quotes given by the policy considers the cumulative quantities available on each aip_entry for the generic products. This policy works well with consume_earlier. The generic products and the specific products could set consume_earlier and expect the policy to give individual quotes considering the cumulative quantities available in those ATP_Entry periods. When there are multiple item_requests in a delivery_request, then the delivery date of this policy's promising_policy_aip would be that of the latest date of all the product_aip's combined. Note that the product_aip's would still have their own 'dates' depending on the buckets in which those quantities were found. The offered quantities would be such that the consumption would be as close to the delivery date as possible.

Finally, if not all the item_requests could be satisfied within the planning horizon, then this policy would not give any quote.

BUCKETED_MIN_PRICE_ASAP -- a promising_policy extension of model Delivery_Request

This policy's behavior would be similar to that of BUCKETED_ALL_MIN_PRICE with some differences. This policy gives promises such that as much of delivery as possible could be made on the due date, and the rest of the quantities could be delivered as soon as possible, if all of the quantities could not be satisfied by the due date. There would be multiple deliveries starting from the due date, if all the requested quantities could not be delivered on the requested date.

For a given item request, the product that could deliver the quantities earliest would be considered. If more than one product could satisfy by the same date, then one of them with the minimum 'price' would be chosen.

This policy works well with generic products. The incremental quotes given by the policy considers the cumulative quantities available on each aip_entry for the generic products.

If you use consume_earlier, then early deliveries could occur. That behavior could change in the later versions of this promising policy.

Extensions	SHIP_IN_RATIO Extension
------------	-------------------------

SHIP_IN_RATIO -- a promising_policy extension of model Delivery_Request

This is similar to ASAP promising policy, except that each delivery would contain all the items requested in the ratio of the quantities of the original delivery request. This policy takes care of generic products, consume, carrier, generic and alternate products shared among the products of the items in the item requests. When such sharing occur, this policy would try to split the available quantity among the items in such a way that the ratio of the items in the delivery could be maintained.

10.31.2 fulfillment_policy extensions of model Delivery_Request

ON_TIME -- a fulfillment_policy extension of model Delivery_Request

If any item in an ON_TIME Delivery_Request can not be delivered on time, then the delivery will be shorted. That is, a partial delivery will be made rather than waiting until a full delivery can be made.

FULL_QUANTITIES_OF_ALL_ITEMS -- a fulfillment_policy extension of model Delivery_Request

If any item in a FULL_QUANTITIES_OF_ALL_ITEMS Delivery_Request is unavailable on the delivery date, then the whole delivery will be delayed rather than making a partial delivery.

UNRESTRICTED -- a fulfillment_policy extension of model Delivery_Request

An UNRESTRICTED Delivery_Request can be shorted, delayed, or both.

Extensions	Promise Extensions
------------	--------------------

10.32 Promise Extensions

10.32.1 delivery_policy extensions of model Promise

FIXED -- a delivery_policy extension of model Promise

Fixed delivery policy

<i>Extensions</i>	<i>Delivery_Promise Extensions</i>
-------------------	------------------------------------

10.33 Delivery_Promise Extensions

10.33.1 fulfillment_policy extensions of model Delivery_Promise

ON_TIME -- *a fulfillment_policy extension of model Delivery_Promise*

If any item in an ON_TIME Delivery_Promise can not be delivered on time, then the delivery will be shorted. That is, a partial delivery will be made rather than waiting until a full delivery can be made.

FULL_QUANTITIES_OF_ALL_ITEMS -- *a fulfillment_policy extension of model Delivery_Promise*

If any item in a FULL_QUANTITIES_OF_ALL_ITEMS Delivery_Promise is unavailable on the delivery date, then the whole delivery will be delayed rather than making a partial delivery.

UNRESTRICTED -- *a fulfillment_policy extension of model Delivery_Promise*

An UNRESTRICTED Delivery_Promise can be shorted, delayed, or both.

<i>Extensions</i>	<i>Horizon Extensions</i>
-------------------	---------------------------

10.34 Horizon Extensions

10.34.1 bucket_spec extensions of model Horizon

ONE -- *a bucket_spec extension of model Horizon*

The ONE bucket_spec Horizon will contain a single bucket for the entire duration.

MONTHS -- *a bucket_spec extension of model Horizon*

The horizon will be broken up on calendar month boundaries.

WEEKS -- *a bucket_spec extension of model Horizon*

The horizon will be broken up into week buckets followed by buckets on calendar month boundaries.

first_day_of_week -- *a Integer field of model WEEKS*
Monday is day 1; Sunday is day 7.

Default: 1

week_buckets -- *a Number field of model WEEKS*

The number of week buckets. After that number of week buckets, monthly buckets begin. If oo, then the entire horizon is broken into weeks.

Default: oo

DAYS -- *a bucket_spec extension of model Horizon*

A DAYS Horizon will be broken up into day buckets followed by buckets on calendar month boundaries.

day_buckets -- *a Number field of model DAYS*

The number of day buckets. After that number of day buckets, monthly buckets begin.

If oo, then the entire horizon is broken into days.

Default: oo

DATES -- *a bucket_spec extension of model Horizon*

<i>Extensions</i>	<i>DATES Extension</i>
-------------------	------------------------

The 'bucket_starts' are maintained in sorted order, and there can be no duplicates. When generating buckets, all Dates that are outside the specified Date_Range are forgotten. And the specified Date_Range_start and 'end' are added to the front and rear of the List (unless already present). Thus, the first bucket will always start at the start of the specified Date_Range and the last bucket will always end at the 'end' of the specified Date_Range.

Each bucket will be a Date_Range that starts at the specified_start Date and ends at the next specified_start Date, which is the start of the next bucket. (A Date_Range includes all Dates up to, but not including its 'end' Date -- thus, consecutive buckets will leave no gaps and will have no overlaps.)

This extension provides lots of flexibility to user in specifying any arbitrary bucket sizes.

The DATES model has these submodels :

Horizon_Bucket_Start.

The DATES model has fields that references these models :

Horizon_Bucket_Start.

bucket_starts -- a list of Horizon_Bucket_Start submodels of model DATES

List of bucket_starts specifying bucket boundaries.

<i>Extensions</i>	<i>Delivery_Acceptance Extensions</i>
-------------------	---------------------------------------

10.35 Delivery_Acceptance Extensions

10.35.1 fulfillment_policy extensions of model Delivery_Acceptance

ON_TIME -- a fulfillment_policy extension of model Delivery_Acceptance

If any item in an ON_TIME Delivery_Acceptance can not be delivered on time, then the delivery will be shorted. That is, a partial delivery will be made rather than waiting until a full delivery can be made.

FULL_QUANTITIES_OF_ALL_ITEMS -- a fulfillment_policy extension of model Delivery_Acceptance

If any item in a FULL_QUANTITIES_OF_ALL_ITEMS Delivery_Acceptance is unavailable on the delivery date, then the whole delivery will be delayed rather than making a partial delivery.

UNRESTRICTED -- a fulfillment_policy extension of model Delivery_Acceptance

An UNRESTRICTED Delivery_Acceptance can be shorted, delayed, or both.

<i>Extensions</i>	<i>Strategy Extensions</i>
-------------------	----------------------------

10.36 Strategy Extensions

10.36.1 termination extensions of model Strategy

NO_PROBLEMS -- a termination extension of model Strategy

A **NO_PROBLEMS** termination Strategy will run until there are no more problems or it finds no improvements to the plan for more than 'max_stable_time' or until it has run in all for more than 'max_run_time'. It specifies no additional termination condition.

problem_filter -- a Expression field of model **NO_PROBLEMS**

An Expression that is passed a Problem as 'w' and should return "true" if the Problem should be counted.

Default: true

TARGET_ACHIEVED -- a termination extension of model Strategy

A **TARGET_ACHIEVED** termination Strategy will run until it achieves the target as specified by the 'goal' extension fields. Note that some 'goal' extensions specify no target, and thus will never be terminated by achieving the target.

RESOLVE_COUNT_EXCEEDED -- a termination extension of model Strategy

A **RESOLVE_COUNT_EXCEEDED** termination Strategy will run until it has attempted to resolve 'max_resolve_count' problems.

MANUAL -- a termination extension of model Strategy

A **MANUAL** termination Strategy will run until user hit done button

10.36.2 execution extensions of model Strategy

SEQUENCE_RUN_ONCE -- a execution extension of model Strategy

A **SEQUENCE_RUN_ONCE** strategy executes it's 'sub_strategies' in a particular sequence once. Each Ordered_Sub_Strategy will be run to completion once in a given order. Sequence is defined using 'sequence' field of Ordered_Sub_Strategy model. The lower the number the earlier the strategy is performed.

The **SEQUENCE_RUN_ONCE** model has these submodels :

Ordered_Sub_Strategy.

The **SEQUENCE_RUN_ONCE** model has fields that references these models :

<i>Extensions</i>	<i>SEQUENCE_RUN_MULTIPLE Extension</i>
-------------------	--

Ordered_Sub_Strategy.

sub_strategies -- a list of Ordered_Sub_Strategy submodels of model

SEQUENCE_RUN_ONCE

An Ordered_Sub_Strategy that will be performed in a given sequence.

SEQUENCE_RUN_MULTIPLE -- a execution extension of model Strategy

A **SEQUENCE_RUN_MULTIPLE** strategy executes it's 'sub_strategies' in a particular sequence. It repeats that sequence until its termination condition is reached. Each Ordered_Sub_Strategy will be run to completion. Sequence is defined using 'sequence' field of Ordered_Sub_Strategy model. The lower the number the earlier the strategy is performed.

The **SEQUENCE_RUN_MULTIPLE** model has these submodels :

Ordered_Sub_Strategy.

The **SEQUENCE_RUN_MULTIPLE** model has fields that references these models :

Ordered_Sub_Strategy.

sub_strategies -- a list of Ordered_Sub_Strategy submodels of model

SEQUENCE_RUN_MULTIPLE

An Ordered_Sub_Strategy that will be performed in a given sequence.

BEFORE_AND_AFTER -- a execution extension of model Strategy

A **BEFORE_AND_AFTER** strategy allows execution of different Strategy's before and after 'run', 'search' and 'resolve' stages of it's owner Strategy run.

A Strategy "run" continues until one of it's termination conditions are met. A "run" consists of a series of "searches". A "search" is a series of Problem "resolves" that result in a new plan. If a search results in a better Plan, where better is defined by the Strategy's 'goals', then the search is termed a "success".

This extension can be used in many different ways. User can setup a 'after_run/search' Strategy that will drive the plan towards feasibility at the end of each 'search' or 'run' or setup a cleanup strategy that will do certain cleanup task(s)(e.g. get rid of excess) after each 'search' or 'run'.

The 'after_resolve' strategy will run after each problem is resolved.

The **BEFORE_AND_AFTER** model has fields that references these models :

Strategy:

before_run -- *a Strategy field of model BEFORE_AND_AFTER*

This Strategy will run before the start of 'owner' Strategy run.

Default: [unspecified]

before_search -- *a Strategy field of model BEFORE_AND_AFTER*

This strategy will run before the start of each search.

Default: [unspecified]

after_resolve -- *a Strategy field of model BEFORE_AND_AFTER*

This strategy will run after each problem is resolved.

Default: [unspecified]

after_search -- *a Strategy field of model BEFORE_AND_AFTER*

This strategy will run after each search is complete.

Default: [unspecified]

after_success -- *a Strategy field of model BEFORE_AND_AFTER*

This Strategy will run after each successful search. It will only run if a search results in a better Plan, where 'better' is defined by the Strategy 'goals', then the search is termed a "success".

Default: [unspecified]

after_run -- *a Strategy field of model BEFORE_AND_AFTER*

This Strategy will run at the completion of 'owner' Strategy run.

Default: [unspecified]

SEQUENTIAL_ALTERNATES -- a execution extension of model Strategy

A SEQUENTIAL_ALTERNATES strategy allows the user to search through a series of alternates. A SEQUENTIAL_ALTERNATES strategy chooses a problem and then iterates from min_all rank to max_all rank. For each rank the resolver is called (restricted by the change category to move to or resize on only alternates with that rank). Then the propagation sub-strategy is called. If the result achieves the goal of the evaluation strategy then the loop is terminated. Else the cleanup sub-strategy is run and we move to the next alternate.

The SEQUENTIAL_ALTERNATES model has fields that references these models : Strategy.

propagation -- *a Strategy field of model SEQUENTIAL_ALTERNATES*

This Strategy will run after each resolve of the parent.

Default: [unspecified]

evaluation -- *a Strategy field of model SEQUENTIAL_ALTERNATES*

The goal function of this strategy is used to evaluate each alternate.

Default: [unspecified]

cleanup -- *a Strategy field of model SEQUENTIAL_ALTERNATES*

This Strategy will run after each failed alternate, before moving to the next alternate.

Default: [unspecified]

min_all -- *a Integer field of model SEQUENTIAL_ALTERNATES*

The min_all and max_all defines the range of alternate operations considered. For example, if min_all = max_all = 0, only the first alternate is allowed.

Default: 0

max_all -- *a Integer field of model SEQUENTIAL_ALTERNATES*

See notes in min_all.

Default: 100

SEQUENTIAL_ALTERNATES_KEEP_BEST -- a execution extension of model Strategy

A SEQUENTIAL_ALTERNATES_KEEP_BEST strategy allows the user to search through a series of alternates and keep the best move under a given

chooses a problem and then iterates from min_all rank to max_all rank. For each rank the resolver is called (restricted by the change category to move to or resize on only alternates with that rank). Then the propagation sub-strategy is called. If the result achieves the goal of the evaluation strategy then the loop is terminated. Else the strategy undoes to the point before the resolver call and moves to the next alternate. Once all alternates have been examined, the strategy returns to the best alternate (as measured by the goal of the evaluation strategy), calls the resolver restricted to that alternate, and then runs the propagation strategy.

The SEQUENTIAL_ALTERNATES_KEEP_BEST model has fields that references these models : Strategy.

propagation -- *a Strategy field of model*

SEQUENTIAL_ALTERNATES_KEEP_BEST

This Strategy will run after each resolve of the parent.

<i>Extensions</i>	<i>PROPORTIONAL_RESOLVES Extension</i>
-------------------	--

Default: [unspecified]

evaluation -- *a Strategy field of model*
SEQUENTIAL_ALTERNATES_KEEP_BEST
The goal function of this strategy is used to evaluate each alternate.
Default: [unspecified]

min_alt -- *a Integer field of model* **SEQUENTIAL_ALTERNATES_KEEP_BEST**
The min_alt and max_alt defines the range of alternate operations considered. For example, if min_alt = max_alt = 0, only the first alternate is allowed.
Default: 0

max_alt -- *a Integer field of model*
SEQUENTIAL_ALTERNATES_KEEP_BEST
See notes in min_alt.
Default: 100

PROPORTIONAL_RESOLVES -- *a execution extension of model Strategy*

A **PROPORTIONAL_RESOLVES** strategy executes it's ranked_sub_strategies' it probabilistically selects a Ranked_Sub_Strategy to perform one problem resolution. The criteria of probabilistic selection is: Ranked_Sub_Strategy.rank * Ranked_Sub_Strategy.strategy.total_interaction

The **PROPORTIONAL_RESOLVES** model has these submodels :

Ranked_Sub_Strategy.

The **PROPORTIONAL_RESOLVES** model has fields that references these models :

Ranked_Sub_Strategy.

ranked_sub_strategies -- *a list of Ranked_Sub_Strategy submodels of model*
PROPORTIONAL_RESOLVES
A Ranked_Sub_Strategy that can be selected and performed by the 'owner' Strategy.

ORDERED_RESOLVES -- *a execution extension of model Strategy*

A **ORDERED_RESOLVES** strategy executes the highest rank Ranked_Sub_Strategy with a problem after each resolve.

The **ORDERED_RESOLVES** model has these submodels :

Ordered_Sub_Strategy.

<i>Extensions</i>	<i>problem_selection extensions of model Strategy</i>
-------------------	---

The **ORDERED_RESOLVES** model has fields that references these models :

Ordered_Sub_Strategy.

sub_strategies -- *a list of Ordered_Sub_Strategy submodels of model*
ORDERED_RESOLVES
A Ranked_Sub_Strategy that can be selected and performed by the 'owner' Strategy.

10.36.3 problem_selection extensions of model Strategy

PROPORTIONAL_INTERACTION -- *a problem_selection extension of model Strategy*

The **PROPORTIONAL_INTERACTION** problem_selection extension selects the problem to solve using following logic.

1. It first selects the Problem_Set probabilistically, weighted by Problem_Set.focus times the total interaction of problems within that Problem_Set. 2. Choose a problem within selected Problem_Set probabilistically, weighted by the interaction value of Problem. Otherwise return the current chosen problem.

At Strategy.hear == 0, it will first select the Problem_Set with the largest Problem_Set.focus value and within that it will select the Problem with the largest interaction value.

When extension's behavior becomes deterministic, it will break the ties by selecting the problem with the earliest startdate. If there are multiple problems with the same start date, it will break the further ties by selecting problem randomly.

EARLIEST_PROBLEM_START -- *a problem_selection extension of model Strategy*

The **EARLIEST_PROBLEM_START** problem_selection extension selects the problem probabilistically, weighted by problem's start date. As the Strategy.hear approaches zero, probability of selecting the problem with the earliest start date increases continuously. At Strategy.selection.hear == 0, it will be completely deterministic and will always select the problem with the earliest start date first.

When the selection becomes deterministic, it will break the tie among the same start date problems, by selecting the problem with the highest interaction. If there are multiple problems with the same interaction value, it will break the further tie by selecting a problem randomly.

SORT_BY_EXPRESSION -- *a problem_selection extension of model Strategy*

Extensions	<i>SORT_BY_EXPRESSION Extension</i>
------------	--

The **SORT_BY_EXPRESSION** problem_selection extension selects the first problem from the sorted problem list. The problem list is sorted using the user specified **sort_criteria** expression that is passed two **Active_Problems** "a" & "b".

Example Usage:

1) strategy.sorting_criteria = "a.problem.dates.start < b.problem.dates.start" The above expression will sort the problem list in an ascending order of problem start dates and this extension will pick the earliest start date problem first.

2) strategy.sorting_criteria = "if (a.problem.dates.start != b.problem.dates.start, a.problem.dates.start < b.problem.dates.start, a.interaction > b.interaction) The above expression specifies Problem start date as a primary key and **Active_Problem.interaction** as a secondary key.

Note, if no **sorting_criteria** expression is specified, problem list will be sorted by problem creation date.

sorting_criteria - - *a Expression field of model SORT_BY_EXPRESSION*
An Expression that is passed two **Active_Problems** "a" & "b". If "a" should be selected before "b", return true, otherwise false.
Default: true

Extensions	<i>Problem_Set Extensions</i>
------------	--------------------------------------

10.37 Problem_Set Extensions

10.37.1 category extensions of model **Problem_Set**

REQUEST_NOT_PLANNED -- *a category extension of model Problem_Set*

A **REQUEST_NOT_PLANNED** **Problem_Set** selects **REQUEST_NOT_PLANNED** Problems with certain characteristics.

This is a subset of the **REQUEST_Problem_Set** category.

item_request_filter - - *a Expression field of model REQUEST_NOT_PLANNED*
An Expression that is passed an **Item_Request** as "a" and should return "true" if the **REQUEST_NOT_PLANNED** Problems with that **Item_Request** should be included in the **Problem_Set**. If it returns "false", then **REQUEST_NOT_PLANNED** Problems with that **Item_Request** will be excluded.
Default: true

problem_filter - - *a Expression field of model REQUEST_NOT_PLANNED*
An Expression that is passed a **Problem** as "a" and should return "true" if the identified **REQUEST_NOT_PLANNED** Problems should be included in the **Problem_Set**. If it returns "false", then the identified Problems will be excluded.
Default: true

REQUEST_PLANNED_LATE -- *a category extension of model Problem_Set*

A **REQUEST_PLANNED_LATE** **Problem_Set** selects **REQUEST_PLANNED_LATE** Problems with certain characteristics.

This is a subset of the **REQUEST** and **REQUEST_PLAN** **Problem_Set** categories.

min_latency - - *a Time field of model REQUEST_PLANNED_LATE*
Problems with less latency than this are excluded.
Default: 0

item_request_filter - - *a Expression field of model REQUEST_PLANNED_LATE*
An Expression that is passed an **Item_Request** as "a" and should return "true" if the **REQUEST_PLANNED_LATE** Problems with that **Item_Request** should be included in the **Problem_Set**. If it returns "false", then **REQUEST_PLANNED_LATE** Problems with that **Item_Request** will be excluded.
Default: true

problem_filter -- *a Expression field of model REQUEST_PLANNED_LATE*
An Expression that is passed a Problem as '#' and should return "true" if the identified REQUEST_PLANNED_LATE Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
Default: true

REQUEST_PLANNED_EARLY -- *a category extension of model Problem_Set*

A REQUEST_PLANNED_EARLY Problem_Set selects REQUEST_PLANNED_EARLY Problems with certain characteristics.

This is a subset of the REQUEST and REQUEST_PLAN Problem_Set categories.

min_earliness -- *a Time field of model REQUEST_PLANNED_EARLY*
Problems with less earliness than this are excluded.
Default: 0

item_request_filter -- *a Expression field of model REQUEST_PLANNED_EARLY*

An Expression that is passed an Item_Request as '#' and should return "true" if the REQUEST_PLANNED_EARLY Problems with that Item_Request should be included in the Problem_Set. If it returns "false", then REQUEST_PLANNED_EARLY Problems with that Item_Request will be excluded.
Default: true

problem_filter -- *a Expression field of model REQUEST_PLANNED_EARLY*
An Expression that is passed a Problem as '#' and should return "true" if the identified REQUEST_PLANNED_EARLY Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
Default: true

REQUEST_PLANNED_SHORT -- *a category extension of model Problem_Set*

A REQUEST_PLANNED_SHORT Problem_Set selects REQUEST_PLANNED_SHORT Problems with certain characteristics.

This is a subset of the REQUEST and REQUEST_PLAN Problem_Set categories.

min_shortness -- *a Quantity field of model REQUEST_PLANNED_SHORT*
Problems with less shortness than this are excluded.
Default: 0

item_request_filter -- *a Expression field of model REQUEST_PLANNED_SHORT*

An Expression that is passed an Item_Request as '#' and should return "true" if the REQUEST_PLANNED_SHORT Problems with that Item_Request should be included in the Problem_Set. If it returns "false", then REQUEST_PLANNED_SHORT Problems with that Item_Request will be excluded.
Default: true

problem_filter -- *a Expression field of model REQUEST_PLANNED_SHORT*
An Expression that is passed a Problem as '#' and should return "true" if the identified REQUEST_PLANNED_SHORT Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

REQUEST_PLANNED_EXCESS -- *a category extension of model Problem_Set*

A REQUEST_PLANNED_EXCESS Problem_Set selects REQUEST_PLANNED_EXCESS Problems with certain characteristics.

This is a subset of the REQUEST and REQUEST_PLAN Problem_Set categories.

min_excess -- *a Quantity field of model REQUEST_PLANNED_EXCESS*
Problems with less than this amount of excess are excluded.
Default: 0

item_request_filter -- *a Expression field of model REQUEST_PLANNED_EXCESS*

An Expression that is passed an Item_Request as '#' and should return "true" if the REQUEST_PLANNED_EXCESS Problems with that Item_Request should be included in the Problem_Set. If it returns "false", then REQUEST_PLANNED_EXCESS Problems with that Item_Request will be excluded.
Default: true

problem_filter -- *a Expression field of model REQUEST_PLANNED_EXCESS*
An Expression that is passed a Problem as '#' and should return "true" if the identified REQUEST_PLANNED_EXCESS Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
Default: true

PROMISE_NOT_PLANNED -- *a category extension of model Problem_Set*

<i>Extensions</i>	<i>PROMISE_PLANNED_LATE Extension</i>
-------------------	---------------------------------------

A **PROMISE_NOT_PLANNED** Problem_Sel selects **PROMISE_NOT_PLANNED** Problems with certain characteristics.

This is a subset of the **PROMISE** Problem_Sel category.

item_promise_filter - - - *a Expression field of model PROMISE_NOT_PLANNED*
An Expression that is passed an Item_Promise as '#' and should return "true" if the **PROMISE_NOT_PLANNED** Problems with that Item_Promise should be included in the Problem_Sel. If it returns "false", then **PROMISE_NOT_PLANNED** Problems with that Item_Promise will be excluded.
Default: true

problem_filter - - - *a Expression field of model PROMISE_NOT_PLANNED*
An Expression that is passed a Problem as '#' and should return "true" if the identified **PROMISE_NOT_PLANNED** Problems should be included in the Problem_Sel. If it returns "false", then the identified Problems will be excluded.
Default: true

PROMISE_PLANNED_LATE - - *a category extension of model Problem_Sel*

A **PROMISE_PLANNED_LATE** Problem_Sel selects **PROMISE_PLANNED_LATE** Problems with certain characteristics.

This is a subset of the **PROMISE** and **PROMISE_PLAN** Problem_Sel categories.

min_lateness - - - *a Time field of model PROMISE_PLANNED_LATE*
Problems with less lateness than this are excluded.
Default: 0

item_promise_filter - - - *a Expression field of model PROMISE_PLANNED_LATE*
An Expression that is passed an Item_Promise as '#' and should return "true" if the **PROMISE_PLANNED_LATE** Problems with that Item_Promise should be included in the Problem_Sel. If it returns "false", then **PROMISE_PLANNED_LATE** Problems with that Item_Promise will be excluded.
Default: true

problem_filter - - - *a Expression field of model PROMISE_PLANNED_LATE*
An Expression that is passed a Problem as '#' and should return "true" if the identified **PROMISE_PLANNED_LATE** Problems should be included in the Problem_Sel. If it returns "false", then the identified Problems will be excluded.
Default: true

<i>Extensions</i>	<i>PROMISE_PLANNED_EARLY Extension</i>
-------------------	--

PROMISE_PLANNED_EARLY - - *a category extension of model Problem_Sel*

A **PROMISE_PLANNED_EARLY** Problem_Sel selects **PROMISE_PLANNED_EARLY** Problems with certain characteristics.

This is a subset of the **PROMISE** and **PROMISE_PLAN** Problem_Sel categories.

min_earliness - - - *a Time field of model PROMISE_PLANNED_EARLY*
Problems with less earliness than this are excluded.
Default: 0

item_promise_filter - - - *a Expression field of model PROMISE_PLANNED_EARLY*
An Expression that is passed an Item_Promise as '#' and should return "true" if the **PROMISE_PLANNED_EARLY** Problems with that Item_Promise should be included in the Problem_Sel. If it returns "false", then **PROMISE_PLANNED_EARLY** Problems with that Item_Promise will be excluded.
Default: true

problem_filter - - - *a Expression field of model PROMISE_PLANNED_EARLY*
An Expression that is passed a Problem as '#' and should return "true" if the identified **PROMISE_PLANNED_EARLY** Problems should be included in the Problem_Sel. If it returns "false", then the identified Problems will be excluded.
Default: true

PROMISE_PLANNED_SHORT - - *a category extension of model Problem_Sel*

A **PROMISE_PLANNED_SHORT** Problem_Sel selects **PROMISE_PLANNED_SHORT** Problems with certain characteristics.

This is a subset of the **PROMISE** and **PROMISE_PLAN** Problem_Sel categories.

min_shortness - - - *a Quantity field of model PROMISE_PLANNED_SHORT*
Problems with less shortness than this are excluded.
Default: 0

<i>Extensions</i>	<i>PROMISE_PLANNED_EXCESS Extension</i>
-------------------	---

Item.promise.filter -- *a Expression field of model*
PROMISE_PLANNED_SHORT
 An Expression that is passed an Item_Promise as '#' and should return "true" if the **PROMISE_PLANNED_SHORT** Problems with that Item_Promise should be included in the Problem_Set. If it returns "false", then **PROMISE_PLANNED_SHORT** Problems with that Item_Promise will be excluded.
 Default: true

problem.filter -- *a Expression field of model* **PROMISE_PLANNED_SHORT**
 An Expression that is passed a Problem as '#' and should return "true" if the identified **PROMISE_PLANNED_SHORT** Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

PROMISE_PLANNED_EXCESS -- *a category extension of model Problem_Set*

A **PROMISE_PLANNED_EXCESS** Problem_Set selects **PROMISE_PLANNED_EXCESS** Problems with certain characteristics.

This is a subset of the **PROMISE** and **PROMISE_PLAN** Problem_Set categories.

min_excess -- *a Quantity field of model* **PROMISE_PLANNED_EXCESS**
 Problems with less excess than this are excluded.
 Default: 0

item.promise.filter -- *a Expression field of model*
PROMISE_PLANNED_EXCESS

An Expression that is passed an Item_Promise as '#' and should return "true" if the **PROMISE_PLANNED_EXCESS** Problems with that Item_Promise should be included in the Problem_Set. If it returns "false", then **PROMISE_PLANNED_EXCESS** Problems with that Item_Promise will be excluded.
 Default: true

problem.filter -- *a Expression field of model* **PROMISE_PLANNED_EXCESS**
 An Expression that is passed a Problem as '#' and should return "true" if the identified **PROMISE_PLANNED_EXCESS** Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

ACCEPTANCE_NOT_PLANNED -- *a category extension of model Problem_Set*

<i>Extensions</i>	<i>ACCEPTANCE_PLANNED_LATE Extension</i>
-------------------	--

A **ACCEPTANCE_NOT_PLANNED** Problem_Set selects **ACCEPTANCE_NOT_PLANNED** Problems with certain characteristics.

This is a subset of the **ACCEPTANCE** Problem_Set category.

item.acceptance.filter -- *a Expression field of model*
ACCEPTANCE_NOT_PLANNED

An Expression that is passed an Item_Acceptance as '#' and should return "true" if the **ACCEPTANCE_NOT_PLANNED** Problems with that Item_Acceptance should be included in the Problem_Set. If it returns "false", then **ACCEPTANCE_NOT_PLANNED** Problems with that Item_Acceptance will be excluded.
 Default: true

problem.filter -- *a Expression field of model* **ACCEPTANCE_NOT_PLANNED**
 An Expression that is passed a Problem as '#' and should return "true" if the identified **ACCEPTANCE_NOT_PLANNED** Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

ACCEPTANCE_PLANNED_LATE -- *a category extension of model Problem_Set*

A **ACCEPTANCE_PLANNED_LATE** Problem_Set selects **ACCEPTANCE_PLANNED_LATE** Problems with certain characteristics.

This is a subset of the **ACCEPTANCE** and **ACCEPTANCE_PLAN** Problem_Set categories.

min_lateness -- *a Time field of model* **ACCEPTANCE_PLANNED_LATE**
 Problems with less lateness than this are excluded.
 Default: 0

item.acceptance.filter -- *a Expression field of model*
ACCEPTANCE_PLANNED_LATE

An Expression that is passed an Item_Acceptance as '#' and should return "true" if the **ACCEPTANCE_PLANNED_LATE** Problems with that Item_Acceptance should be included in the Problem_Set. If it returns "false", then **ACCEPTANCE_PLANNED_LATE** Problems with that Item_Acceptance will be excluded.
 Default: true

problem_filter -- *a Expression field of model* ACCEPTANCE_PLANNED_LATE
 An Expression that is passed a Problem as *#* and should return "true" if the identified ACCEPTANCE_PLANNED_LATE Problems should be included in the Problem_Set.
 If it returns "false", then the identified Problems will be excluded.
 Default: true

ACCEPTANCE_PLANNED_EARLY -- *a category extension of model Problem_Set*
 A ACCEPTANCE_PLANNED_EARLY Problem_Set selects
 ACCEPTANCE_PLANNED_EARLY Problems with certain characteristics.

This is a subset of the ACCEPTANCE and ACCEPTANCE_PLAN Problem_Set categories.
 Default: true

min_earliness -- *a Time field of model* ACCEPTANCE_PLANNED_EARLY
 Problems with less earliness than this are excluded.
 Default: 0

item_acceptance_filter -- *a Expression field of model*
 ACCEPTANCE_PLANNED_EARLY
 An Expression that is passed an Item_Acceptance as *#* and should return "true" if the ACCEPTANCE_PLANNED_EARLY Problems with that Item_Acceptance should be included in the Problem_Set. If it returns "false", then
 ACCEPTANCE_PLANNED_EARLY Problems with that Item_Acceptance will be excluded.
 Default: true

problem_filter -- *a Expression field of model*
 ACCEPTANCE_PLANNED_EARLY
 An Expression that is passed a Problem as *#* and should return "true" if the identified ACCEPTANCE_PLANNED_EARLY Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

ACCEPTANCE_PLANNED_SHORT -- *a category extension of model Problem_Set*
 A ACCEPTANCE_PLANNED_SHORT Problem_Set selects
 ACCEPTANCE_PLANNED_SHORT Problems with certain characteristics.
 This is a subset of the ACCEPTANCE and ACCEPTANCE_PLAN Problem_Set categories.
 Default: true

min_shortness -- *a Quantity field of model* ACCEPTANCE_PLANNED_SHORT
 Problems with less shortness than this are excluded.
 Default: 0

item_acceptance_filter -- *a Expression field of model*
 ACCEPTANCE_PLANNED_SHORT
 An Expression that is passed an Item_Acceptance as *#* and should return "true" if the ACCEPTANCE_PLANNED_SHORT Problems with that Item_Acceptance should be included in the Problem_Set. If it returns "false", then
 ACCEPTANCE_PLANNED_SHORT Problems with that Item_Acceptance will be excluded.
 Default: true

problem_filter -- *a Expression field of model*
 ACCEPTANCE_PLANNED_SHORT
 An Expression that is passed a Problem as *#* and should return "true" if the identified ACCEPTANCE_PLANNED_SHORT Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

ACCEPTANCE_PLANNED_EXCESS -- *a category extension of model Problem_Set*
 A ACCEPTANCE_PLANNED_EXCESS Problem_Set selects
 ACCEPTANCE_PLANNED_EXCESS Problems with certain characteristics.

This is a subset of the ACCEPTANCE and ACCEPTANCE_PLAN Problem_Set categories.
 Default: true

min_excess -- *a Quantity field of model* ACCEPTANCE_PLANNED_EXCESS
 Problems with less than this amount of excess are excluded.
 Default: 0

item_acceptance_filter -- *a Expression field of model*
 ACCEPTANCE_PLANNED_EXCESS
 An Expression that is passed an Item_Acceptance as *#* and should return "true" if the ACCEPTANCE_PLANNED_EXCESS Problems with that Item_Acceptance should be included in the Problem_Set. If it returns "false", then
 ACCEPTANCE_PLANNED_EXCESS Problems with that Item_Acceptance will be excluded.
 Default: true

Extensions	PLANNED_BEFORE_CURRENT Extension
------------	----------------------------------

problem_filter -- *a Expression field of model*
ACCEPTANCE_PLANNED_EXCESS
 An Expression that is passed a Problem as "P" and should return "true" if the identified **ACCEPTANCE_PLANNED_EXCESS** Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

PLANNED_BEFORE_CURRENT -- a category extension of model Problem_Set

A **PLANNED_BEFORE_CURRENT** Problem_Set selects **PLANNED_BEFORE_CURRENT** Problems on certain Operations.

operation_plan_filter -- *a Expression field of model*
PLANNED_BEFORE_CURRENT
 An Expression that is passed an Operation_Plan as "P" and should return "true" if the Problems on that Operation should be included in the Problem_Set. If it returns "false", then Problems on that Operation will be excluded.
 Default: true

problem_filter -- *a Expression field of model* **PLANNED_BEFORE_CURRENT**
 An Expression that is passed a Problem as "P" and should return "true" if the identified **PLANNED_BEFORE_CURRENT** Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

UNRELEASED -- a category extension of model Problem_Set

An **UNRELEASED** Problem_Set selects **UNRELEASED** Problems on certain Operations.

operation_plan_filter -- *a Expression field of model* **UNRELEASED**
 An Expression that is passed an Operation_Plan as "P" and should return "true" if the Problems on that Operation should be included in the Problem_Set. If it returns "false", then Problems on that Operation will be excluded.
 Default: true

problem_filter -- *a Expression field of model* **UNRELEASED**
 An Expression that is passed a Problem as "P" and should return "true" if the identified **UNRELEASED** Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

Extensions	NEEDS_RELEASE Extension
------------	-------------------------

NEEDS_RELEASE -- a category extension of model Problem_Set

A **NEEDS_RELEASE** Problem_Set selects **NEEDS_RELEASE** Problems on certain Operations.

operation_plan_filter -- *a Expression field of model* **NEEDS_RELEASE**
 An Expression that is passed an Operation_Plan as "P" and should return "true" if the Problems on that Operation should be included in the Problem_Set. If it returns "false", then Problems on that Operation will be excluded.
 Default: true

problem_filter -- *a Expression field of model* **NEEDS_RELEASE**
 An Expression that is passed a Problem as "P" and should return "true" if the identified **NEEDS_RELEASE** Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

INCONSISTENT_OPPLAN -- a category extension of model Problem_Set

A **INCONSISTENT_OPPLAN** Problem_Set selects **INCONSISTENT_OPPLAN** Problems on certain Operations.

operation_plan_filter -- *a Expression field of model* **INCONSISTENT_OPPLAN**
 An Expression that is passed an Operation_Plan as "P" and should return "true" if the Problems on that Operation should be included in the Problem_Set. If it returns "false", then Problems on that Operation will be excluded.
 Default: true

problem_filter -- *a Expression field of model* **INCONSISTENT_OPPLAN**
 An Expression that is passed a Problem as "P" and should return "true" if the identified **INCONSISTENT_OPPLAN** Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

OPERATION -- a category extension of model Problem_Set

A **OPERATION** Problem_Set selects **OPERATION** Problems on certain Operations.

Extensions	REQUEST_PROMISED_LATE Extension
------------	---------------------------------

operation_plan_filter -- *a Expression field of model OPERATION*
 An Expression that is passed an Operation_Plan as '#' and should return "true" if the Problems on that Operation should be included in the Problem_Set. If it returns "false", then Problems on that Buffer will be excluded.
 Default: true

problem_filter -- *a Expression field of model OPERATION*
 An Expression that is passed a Problem as '#' and should return "true" if the identified OPERATION Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

REQUEST_PROMISED_LATE -- *a category extension of model Problem_Set*

A REQUEST_PROMISED_LATE Problem_Set selects REQUEST_PROMISED_LATE Problems with certain characteristics.

This is a subset of the REQUEST_PROMISE, and REQUEST_PROMISE Problem_Set categories.

min_lateness -- *a Time field of model REQUEST_PROMISED_LATE*
 Problems with less lateness than this are excluded.
 Default: 0

delivery_promise_filter -- *a Expression field of model REQUEST_PROMISED_LATE*
 An Expression that is passed a Delivery_Promise as '#' and should return "true" if the REQUEST_PROMISED_LATE Problems with that Delivery_Promise should be included in the Problem_Set. If it returns "false", then REQUEST_PROMISED_LATE Problems with that Delivery_Promise will be excluded.
 Default: true

problem_filter -- *a Expression field of model REQUEST_PROMISED_LATE*
 An Expression that is passed a Problem as '#' and should return "true" if the identified REQUEST_PROMISED_LATE Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

REQUEST_PROMISED_EARLY -- *a category extension of model Problem_Set*

Extensions	REQUEST_PROMISED_SHORT Extension
------------	----------------------------------

A REQUEST_PROMISED_EARLY Problem_Set selects REQUEST_PROMISED_EARLY Problems with certain characteristics.
 This is a subset of the REQUEST_PROMISE, and REQUEST_PROMISE Problem_Set categories.

min_earliness -- *a Time field of model REQUEST_PROMISED_EARLY*
 Problems with less earliness than this are excluded.
 Default: 0

delivery_promise_filter -- *a Expression field of model REQUEST_PROMISED_EARLY*
 An Expression that is passed a Delivery_Promise as '#' and should return "true" if the REQUEST_PROMISED_EARLY Problems with that Delivery_Promise should be included in the Problem_Set. If it returns "false", then REQUEST_PROMISED_EARLY Problems with that Delivery_Promise will be excluded.
 Default: true

problem_filter -- *a Expression field of model REQUEST_PROMISED_EARLY*
 An Expression that is passed a Problem as '#' and should return "true" if the identified REQUEST_PROMISED_EARLY Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

REQUEST_PROMISED_SHORT -- *a category extension of model Problem_Set*

A REQUEST_PROMISED_SHORT Problem_Set selects REQUEST_PROMISED_SHORT Problems with certain characteristics.

This is a subset of the REQUEST_PROMISE, and REQUEST_PROMISE Problem_Set categories.

min_shortness -- *a Quantity field of model REQUEST_PROMISED_SHORT*
 Problems with less shortness than this are excluded.
 Default: 0

<i>Extensions</i>	<i>REQUEST_PROMISED_EXCESS Extension</i>
-------------------	--

item_promise_filter -- *a Expression field of model*
REQUEST_PROMISED_SHORT
 An Expression that is passed an Item_Promise as '#' and should return "true" if the **REQUEST_PROMISED_SHORT** Problems with that Item_Promise should be included in the Problem_Set. If it returns "false", then **REQUEST_PROMISED_SHORT** Problems with that Item_Promise will be excluded.
 Default: true

problem_filter -- *a Expression field of model* **REQUEST_PROMISED_SHORT**
 An Expression that is passed a Problem as '#' and should return "true" if the identified **REQUEST_PROMISED_SHORT** Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

REQUEST_PROMISED_EXCESS -- *a category extension of model Problem_Set*
 A **REQUEST_PROMISED_EXCESS** Problem_Set selects **REQUEST_PROMISED_EXCESS** Problems with certain characteristics.

This is a subset of the **REQUEST_PROMISE**, and **REQUEST_PROMISE** Problem_Set categories.

min_excess -- *a Quantity field of model* **REQUEST_PROMISED_EXCESS**
 Problems with less excess than this are excluded.
 Default: 0

item_promise_filter -- *a Expression field of model*
REQUEST_PROMISED_EXCESS
 An Expression that is passed an Item_Promise as '#' and should return "true" if the **REQUEST_PROMISED_EXCESS** Problems with that Item_Promise should be included in the Problem_Set. If it returns "false", then **REQUEST_PROMISED_EXCESS** Problems with that Item_Promise will be excluded.
 Default: true

problem_filter -- *a Expression field of model* **REQUEST_PROMISED_EXCESS**
 An Expression that is passed a Problem as '#' and should return "true" if the identified **REQUEST_PROMISED_EXCESS** Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

<i>Extensions</i>	<i>PROMISE_NOT_OFFERED Extension</i>
-------------------	--------------------------------------

PROMISE_NOT_OFFERED -- *a category extension of model Problem_Set*
 A **PROMISE_NOT_OFFERED** Problem_Set selects **PROMISE_NOT_OFFERED** Problems with certain characteristics.

This is a subset of the **REQUEST** and **PROMISE** Problem_Set categories.

request_filter -- *a Expression field of model* **PROMISE_NOT_OFFERED**
 An Expression that is passed a Request as '#' and should return "true" if the Problems with that Request should be included in the Problem_Set. If it returns "false", then Problems with that Request will be excluded.
 Default: true

problem_filter -- *a Expression field of model* **PROMISE_NOT_OFFERED**
 An Expression that is passed a Problem as '#' and should return "true" if the identified **PROMISE_NOT_OFFERED** Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

PROMISE_NOT_ACCEPTED -- *a category extension of model Problem_Set*

A **PROMISE_NOT_ACCEPTED** Problem_Set selects **PROMISE_NOT_ACCEPTED** Problems with certain characteristics.

This is a subset of the **REQUEST** and **PROMISE** Problem_Set categories.

request_filter -- *a Expression field of model* **PROMISE_NOT_ACCEPTED**
 An Expression that is passed a Request as '#' and should return "true" if the Problems with that Request should be included in the Problem_Set. If it returns "false", then Problems with that Request will be excluded.
 Default: true

problem_filter -- *a Expression field of model* **PROMISE_NOT_ACCEPTED**
 An Expression that is passed a Problem as '#' and should return "true" if the identified **PROMISE_NOT_ACCEPTED** Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

ACCEPTANCE_INCONSISTENT -- *a category extension of model Problem_Set*

<i>Extensions</i>	<i>REQUEST_QUEUED Extension</i>
-------------------	---------------------------------

A **ACCEPTANCE_INCONSISTENT** Problem_Set selects **ACCEPTANCE_INCONSISTENT** Problems with certain characteristics.

This is a subset of the **REQUEST** and **PROMISE** Problem_Set categories.

request_filter -- *a Expression field of model* **ACCEPTANCE_INCONSISTENT**
An Expression that is passed a Request as *#* and should return "true" if the Problems with that Request should be included in the Problem_Set. If it returns "false", then Problems with that Request will be excluded.
Default: true

problem_filter -- *a Expression field of model* **ACCEPTANCE_INCONSISTENT**
An Expression that is passed a Problem as *#* and should return "true" if the identified **ACCEPTANCE_INCONSISTENT** Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
Default: true

REQUEST_QUEUED -- *a category extension of model* Problem_Set

A **REQUEST_QUEUED** Problem_Set selects **REQUEST_QUEUED** Problems with certain characteristics.

This is a subset of the **REQUEST** Problem_Set category.

request_filter -- *a Expression field of model* **REQUEST_QUEUED**
An Expression that is passed a Request as *#* and should return "true" if the Problems with that Request should be included in the Problem_Set. If it returns "false", then Problems with that Request will be excluded.
Default: true

problem_filter -- *a Expression field of model* **REQUEST_QUEUED**
An Expression that is passed a Problem as *#* and should return "true" if the identified **REQUEST_QUEUED** Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
Default: true

REQUEST -- *a category extension of model* Problem_Set

A **REQUEST** Problem_Set selects Request Problems (e.g., **REQUEST_PLANNED_LATE**, etc.) on certain **REQUESTS**.

<i>Extensions</i>	<i>REQUEST_PLAN Extension</i>
-------------------	-------------------------------

request_filter -- *a Expression field of model* **REQUEST**
An Expression that is passed a Request as *#* and should return "true" if the Problems with that Request should be included in the Problem_Set. If it returns "false", then Problems with that Request will be excluded.
Default: true

problem_filter -- *a Expression field of model* **REQUEST**
An Expression that is passed a Problem as *#* and should return "true" if the identified **REQUEST** Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
Default: true

REQUEST_PLAN -- *a category extension of model* Problem_Set

A **REQUEST_PLAN** Problem_Set selects Problems between Item_Requests and their delivery plan, including **REQUEST_PLANNED_LATE**, **REQUEST_PLANNED_EARLY**, **REQUEST_PLANNED_SHORT**, and **REQUEST_PLANNED_EXCESS**.

This is a subset of the **REQUEST** Problem_Set category.

item_request_filter -- *a Expression field of model* **REQUEST_PLAN**
An Expression that is passed a Item_Request as *#* and should return "true" if the Problems with that Item_Request should be included in the Problem_Set. If it returns "false", then Problems with that Item_Request will be excluded.
Default: true

problem_filter -- *a Expression field of model* **REQUEST_PLAN**
An Expression that is passed a Problem as *#* and should return "true" if the identified **REQUEST_PLAN** Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
Default: true

PROMISE -- *a category extension of model* Problem_Set

A **PROMISE** Problem_Set selects Promise Problems.

promise_filter -- *a Expression field of model* **PROMISE**
An Expression that is passed a Promise as *#* and should return "true" if the Problems with that Promise should be included in the Problem_Set. If it returns "false", then Problems with that Promise will be excluded.
Default: true

Extensions	PROMISE_PLAN Extension
------------	------------------------

problem_filter - - *a Expression field of model PROMISE*
 An Expression that is passed a Problem as '#' and should return "true" if the identified PROMISE Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

PROMISE_PLAN - - *a category extension of model Problem_Set*

A PROMISE_PLAN Problem_Set selects Problems between Item_Promises and their delivery plan, including PROMISE_PLANNED_LATE, PROMISE_PLANNED_EARLY, PROMISE_PLANNED_SHORT, and PROMISE_PLANNED_EXCESS.

This is a subset of the PROMISE Problem_Set category.

Item_Promise_Filter - - *a Expression field of model PROMISE_PLAN*
 An Expression that is passed a Item_Promise as '#' and should return "true" if the Problems with that Item_Promise should be included in the Problem_Set. If it returns "false", then Problems with that Item_Promise will be excluded.
 Default: true

problem_filter - - *a Expression field of model PROMISE_PLAN*
 An Expression that is passed a Problem as '#' and should return "true" if the identified PROMISE_PLAN Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

REQUEST_PROMISE - - *a category extension of model Problem_Set*

A REQUEST_PROMISE_PLAN Problem_Set selects Problems that result from discrepancies between a Request and the Promise offered to it, including REQUEST_PROMISED_LATE, REQUEST_PROMISED_EARLY, REQUEST_PROMISED_SHORT, REQUEST_PROMISED_EXCESS, DELIVERY_REQUEST_OVERRICED, and ITEM_REQUEST_OVERRICED.

This is a subset of both the REQUEST and PROMISE Problem_Set categories.

delivery_promise_filter - - *a Expression field of model REQUEST_PROMISE*
 An Expression that is passed a Delivery_Promise as '#' and should return "true" if the Problems with that Delivery_Promise should be included in the Problem_Set. If it returns "false", then Problems with that Delivery_Promise will be excluded.

Extensions	DELIVERY_REQUEST_NOT_COORDINATED Extension
------------	--

Default: true

problem_filter - - *a Expression field of model REQUEST_PROMISE*
 An Expression that is passed a Problem as '#' and should return "true" if the identified REQUEST_PROMISE Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

DELIVERY_REQUEST_NOT_COORDINATED - - *a category extension of model Problem_Set*

A DELIVERY_REQUEST_NOT_COORDINATED Problem_Set selects Problems that result from items of a Delivery_Request not all being scheduled for delivery on the same date.

problem_filter - - *a Expression field of model DELIVERY_REQUEST_NOT_COORDINATED*
 An Expression that is passed a Problem as '#' and should return "true" if the identified DELIVERY_REQUEST_NOT_COORDINATED Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

DELIVERY_PROMISE_NOT_COORDINATED - - *a category extension of model Problem_Set*

A DELIVERY_PROMISE_NOT_COORDINATED Problem_Set selects Problems that result from items of a Delivery_Promise not all being scheduled for delivery on the same date.

problem_filter - - *a Expression field of model DELIVERY_PROMISE_NOT_COORDINATED*
 An Expression that is passed a Problem as '#' and should return "true" if the identified DELIVERY_PROMISE_NOT_COORDINATED Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

DELIVERY_ACCEPTANCE_NOT_COORDINATED - - *a category extension of model Problem_Set*

A DELIVERY_ACCEPTANCE_NOT_COORDINATED Problem_Set selects Problems that result from items of a Delivery_Acceptance not all being scheduled for delivery on the same date.

problem_filter -- *a Expression field of model*
DELIVERY_ACCEPTANCE_NOT_COORDINATED
An Expression that is passed a Problem as # and should return "true" if the identified DELIVERY_ACCEPTANCE_NOT_COORDINATED Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
Default: true

SUPPLY_PLANNED_LATE -- *a category extension of model Problem_Set*
A SUPPLY_PLANNED_LATE Problem_Set selects SUPPLY_PLANNED_LATE Problems with certain characteristics.

This is a subset of the SUPPLY and SUPPLY_PLAN Problem_Set categories.

min_lateness -- *a Time field of model SUPPLY_PLANNED_LATE*
Problems with less lateness than this are excluded.
Default: 0

item_request_filter -- *a Expression field of model SUPPLY_PLANNED_LATE*
An Expression that is passed an Item_Request as # and should return "true" if the SUPPLY_PLANNED_LATE Problems with that Item_Request should be included in the Problem_Set. If it returns "false", then SUPPLY_PLANNED_LATE Problems with that Item_Request will be excluded.
Default: true

problem_filter -- *a Expression field of model SUPPLY_PLANNED_LATE*
An Expression that is passed a Problem as # and should return "true" if the identified SUPPLY_PLANNED_LATE Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
Default: true

SUPPLY_PLANNED_EARLY -- *a category extension of model Problem_Set*

A SUPPLY_PLANNED_EARLY Problem_Set selects SUPPLY_PLANNED_EARLY Problems with certain characteristics.

This is a subset of the SUPPLY and SUPPLY_PLAN Problem_Set categories.

min_earliness -- *a Time field of model SUPPLY_PLANNED_EARLY*
Problems with less earliness than this are excluded.
Default: 0

item_request_filter -- *a Expression field of model SUPPLY_PLANNED_EARLY*
An Expression that is passed an Item_Request as # and should return "true" if the SUPPLY_PLANNED_EARLY Problems with that Item_Request should be included in the Problem_Set. If it returns "false", then SUPPLY_PLANNED_EARLY Problems with that Item_Request will be excluded.
Default: true

problem_filter -- *a Expression field of model SUPPLY_PLANNED_EARLY*
An Expression that is passed a Problem as # and should return "true" if the identified SUPPLY_PLANNED_EARLY Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
Default: true

SUPPLY_PLANNED_SHORT -- *a category extension of model Problem_Set*

A SUPPLY_PLANNED_SHORT Problem_Set selects SUPPLY_PLANNED_SHORT Problems with certain characteristics.

This is a subset of the SUPPLY and SUPPLY_PLAN Problem_Set categories.

min_shortness -- *a Quantity field of model SUPPLY_PLANNED_SHORT*
Problems with less shortness than this are excluded.
Default: 0

item_request_filter -- *a Expression field of model SUPPLY_PLANNED_SHORT*
An Expression that is passed an Item_Request as # and should return "true" if the SUPPLY_PLANNED_SHORT Problems with that Item_Request should be included in the Problem_Set. If it returns "false", then SUPPLY_PLANNED_SHORT Problems with that Item_Request will be excluded.
Default: true

problem_filter -- *a Expression field of model SUPPLY_PLANNED_SHORT*
An Expression that is passed a Problem as # and should return "true" if the identified SUPPLY_PLANNED_SHORT Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
Default: true

SUPPLY_PLANNED_EXCESS -- *a category extension of model Problem_Set*

A SUPPLY_PLANNED_EXCESS Problem_Set selects SUPPLY_PLANNED_EXCESS Problems with certain characteristics.

This is a subset of the SUPPLY and SUPPLY_PLAN Problem_Set categories.

min_excess -- *a Quantity field of model SUPPLY_PLANNED_EXCESS*
Problems with less than this amount of excess are excluded.

Default: 0

item_request_filter -- *a Expression field of model SUPPLY_PLANNED_EXCESS*
An Expression that is passed an Item_Request as # and should return "true" if the SUPPLY_PLANNED_EXCESS Problems with that Item_Request should be included in the Problem_Set. If it returns "false", then SUPPLY_PLANNED_EXCESS Problems with that Item_Request will be excluded.

Default: true

problem_filter -- *a Expression field of model SUPPLY_PLANNED_EXCESS*

An Expression that is passed a Problem as # and should return "true" if the identified SUPPLY_PLANNED_EXCESS Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

SUPPLY_PROMISED_LATE -- *a category extension of model Problem_Set*

A SUPPLY_PROMISED_LATE Problem_Set selects SUPPLY_PROMISED_LATE Problems with certain characteristics.

This is a subset of the SUPPLY and SUPPLY_PROMISE Problem_Set categories.

min_lateness -- *a Time field of model SUPPLY_PROMISED_LATE*
Problems with less lateness than this are excluded.

Default: 0

delivery_promise_filter -- *a Expression field of model*

SUPPLY_PROMISED_LATE

An Expression that is passed an Delivery_Promise as # and should return "true" if the SUPPLY_PROMISED_LATE Problems with that Delivery_Promise should be included in the Problem_Set. If it returns "false", then SUPPLY_PROMISED_LATE Problems with that Delivery_Promise will be excluded.

Default: true

problem_filter -- *a Expression field of model SUPPLY_PROMISED_LATE*
An Expression that is passed a Problem as # and should return "true" if the identified SUPPLY_PROMISED_LATE Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

SUPPLY_PROMISED_EARLY -- *a category extension of model Problem_Set*

A SUPPLY_PROMISED_EARLY Problem_Set selects SUPPLY_PROMISED_EARLY Problems with certain characteristics.

This is a subset of the SUPPLY and SUPPLY_PROMISE Problem_Set categories.

min_earliness -- *a Time field of model SUPPLY_PROMISED_EARLY*
Problems with less earliness than this are excluded.

Default: 0

delivery_promise_filter -- *a Expression field of model*

SUPPLY_PROMISED_EARLY

An Expression that is passed an Delivery_Promise as # and should return "true" if the SUPPLY_PROMISED_EARLY Problems with that Delivery_Promise should be included in the Problem_Set. If it returns "false", then SUPPLY_PROMISED_EARLY Problems with that Delivery_Promise will be excluded.

Default: true

problem_filter -- *a Expression field of model SUPPLY_PROMISED_EARLY*
An Expression that is passed a Problem as # and should return "true" if the identified SUPPLY_PROMISED_EARLY Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

SUPPLY_PROMISED_SHORT -- *a category extension of model Problem_Set*

A SUPPLY_PROMISED_SHORT Problem_Set selects SUPPLY_PROMISED_SHORT Problems with certain characteristics.

This is a subset of the SUPPLY and SUPPLY_PROMISE Problem_Set categories.

min_shortness -- *a Quantity field of model SUPPLY_PROMISED_SHORT*
Problems with less shortness than this are excluded.

Default: 0

<i>Extensions</i>	<i>SUPPLY_PROMISED_EXCESS Extension</i>
-------------------	---

item_promise_filter -- *a Expression field of model*
SUPPLY_PROMISED_SHORT
 An Expression that is passed an Item_Promise as *#* and should return "true" if the SUPPLY_PROMISED_SHORT Problems with that Item_Promise should be included in the Problem_Set. If it returns "false", then SUPPLY_PROMISED_SHORT Problems with that Item_Promise will be excluded.
 Default: true

problem_filter -- *a Expression field of model* SUPPLY_PROMISED_SHORT
 An Expression that is passed a Problem as *#* and should return "true" if the identified SUPPLY_PROMISED_SHORT Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

SUPPLY_PROMISED_EXCESS -- *a category extension of model Problem_Set*

A SUPPLY_PROMISED_EXCESS Problem_Set selects SUPPLY_PROMISED_EXCESS Problems with certain characteristics.

This is a subset of the SUPPLY and SUPPLY_PROMISE Problem_Set categories.

min_excess -- *a Quantity field of model* SUPPLY_PROMISED_EXCESS
 Problems with less excess than this are excluded.
 Default: 0

item_promise_filter -- *a Expression field of model* SUPPLY_PROMISED_EXCESS

An Expression that is passed an Item_Promise as *#* and should return "true" if the SUPPLY_PROMISED_EXCESS Problems with that Item_Promise should be included in the Problem_Set. If it returns "false", then SUPPLY_PROMISED_EXCESS Problems with that Item_Promise will be excluded.
 Default: true

problem_filter -- *a Expression field of model* SUPPLY_PROMISED_EXCESS
 An Expression that is passed a Problem as *#* and should return "true" if the identified SUPPLY_PROMISED_EXCESS Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

SUPPLY -- *a category extension of model Problem_Set*

<i>Extensions</i>	<i>SUPPLY_PLAN Extension</i>
-------------------	------------------------------

A SUPPLY Problem_Set selects Supply Request Problems (e.g., SUPPLY_PLANNED_LATE, etc.) on certain supply_requests.

request_filter -- *a Expression field of model* SUPPLY
 An Expression that is passed a Request as *#* and should return "true" if the Problems with that Request should be included in the Problem_Set. If it returns "false", then Problems with that Request will be excluded.
 Default: true

problem_filter -- *a Expression field of model* SUPPLY
 An Expression that is passed a Problem as *#* and should return "true" if the identified SUPPLY Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

SUPPLY_PLAN -- *a category extension of model Problem_Set*

A SUPPLY_PLAN Problem_Set selects Problems between the receiving_plan and the delivery_plan of supply Requests, including SUPPLY_PLANNED_LATE, SUPPLY_PLANNED_EARLY, SUPPLY_PLANNED_SHORT, and SUPPLY_PLANNED_EXCESS.

This is a subset of the SUPPLY Problem_Set category.

item_request_filter -- *a Expression field of model* SUPPLY_PLAN

An Expression that is passed a Item_Request as *#* and should return "true" if the Problems with that Item_Request should be included in the Problem_Set. If it returns "false", then Problems with that Item_Request will be excluded.
 Default: true

problem_filter -- *a Expression field of model* SUPPLY_PLAN
 An Expression that is passed a Problem as *#* and should return "true" if the identified SUPPLY_PLAN Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

SUPPLY_PROMISE -- *a category extension of model Problem_Set*

A SUPPLY_PROMISE_PLAN Problem_Set selects Problems that result from discrepancies between the receiving_plan of a supply Request's and the Promise offered to it, including SUPPLY_PROMISED_LATE, SUPPLY_PROMISED_EARLY, SUPPLY_PROMISED_SHORT, and SUPPLY_PROMISED_EXCESS.

<i>Extensions</i>	<i>UNIDENTIFIED_OP_STATE Extension</i>
-------------------	--

This is a subset of the SUPPLY Problem_Sel category.

delivery_promise_filter -- *a Expression field of model SUPPLY_PROMISE*
 An Expression that is passed a Delivery_Promise as "#" and should return "true" if the Problems with that Delivery_Promise should be included in the Problem_Sel. If it returns "false", then Problems with that Delivery_Promise will be excluded.
 Default: true

problem_filter -- *a Expression field of model SUPPLY_PROMISE*
 An Expression that is passed a Problem as "#" and should return "true" if the identified SUPPLY_PROMISE Problems should be included in the Problem_Sel. If it returns "false", then the identified Problems will be excluded.
 Default: true

UNIDENTIFIED_OP_STATE -- *a category extension of model Problem_Set*
 A UNIDENTIFIED_OP_STATE Problem_Set selects UNIDENTIFIED_OP_STATE Problems on certain Operations.

operation_plan_filter -- *a Expression field of model UNIDENTIFIED_OP_STATE*
 An Expression that is passed an Operation_Plan as "#" and should return "true" if the Problems on that Operation should be included in the Problem_Sel. If it returns "false", then Problems on that Operation will be excluded.
 Default: true

problem_filter -- *a Expression field of model UNIDENTIFIED_OP_STATE*
 An Expression that is passed a Problem as "#" and should return "true" if the identified UNIDENTIFIED_OP_STATE Problems should be included in the Problem_Sel. If it returns "false", then the identified Problems will be excluded.
 Default: true

UNCONSOLIDATED -- *a category extension of model Problem_Set*
 Unconsolidated Operation_Plans exist in this resource_plan.

resource_plan_filter -- *a Expression field of model UNCONSOLIDATED*
 An Expression that is passed a Resource_Plan as "#" and should return "true" if the UNCONSOLIDATED Problems on that Resource should be included in the Problem_Sel. If it returns "false", then Problems on that Resource will be excluded.
 Default: true

<i>Extensions</i>	<i>UNCOORDINATED Extension</i>
-------------------	--------------------------------

UNCOORDINATED -- *a category extension of model Problem_Set*

Uncoordinated Operation_Plans exist in this resource_plan.

resource_plan_filter -- *a Expression field of model UNCOORDINATED*
 An Expression that is passed a Resource_Plan as "#" and should return "true" if the UNCOORDINATED Problems on that Resource should be included in the Problem_Sel. If it returns "false", then Problems on that Resource will be excluded.
 Default: true

CONSOLIDATION_OVERSIZE -- *a category extension of model Problem_Set*
 CONSOLIDATION_OVERSIZED Operation_Plans exist in this resource_plan.

resource_plan_filter -- *a Expression field of model CONSOLIDATION_OVERSIZE*
 An Expression that is passed a Resource_Plan as "#" and should return "true" if the CONSOLIDATION_OVERSIZED Problems on that Resource should be included in the Problem_Sel. If it returns "false", then Problems on that Resource will be excluded.
 Default: true

CONSOLIDATION_UNDERSIZE -- *a category extension of model Problem_Set*
 CONSOLIDATION_UNDERIZED Operation_Plans exist in this resource_plan.

resource_plan_filter -- *a Expression field of model CONSOLIDATION_UNDERSIZE*
 An Expression that is passed a Resource_Plan as "#" and should return "true" if the CONSOLIDATION_UNDERIZED Problems on that Resource should be included in the Problem_Sel. If it returns "false", then Problems on that Resource will be excluded.
 Default: true

OVERLOAD -- *a category extension of model Problem_Set*

An OVERLOAD Problem_Set selects OVERLOAD Problems with certain characteristics.

resource_plan_filter -- *a Expression field of model OVERLOAD*
 An Expression that is passed a Resource_Plan as "#" and should return "true" if the OVERLOAD Problems on that Resource should be included in the Problem_Sel.

Default: true

problem_filter -- *a Expression field of model OVERLOAD*

An Expression that is passed a Problem as '#' and should return "true" if the identified OVERLAP Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

min_overload_time -- *a Time field of model OVERLOAD*

A Problem will be excluded if its 'overload_time' is less than the minimum Time specified by this Time_Horizon for the 'dates.start' of the Problem. The default, 0, will not exclude any OVERLOAD Problem.

Default: 0

min_overload_std_time -- *a Time field of model OVERLOAD*

A Problem will be excluded if its 'overload_std_time' is less than the minimum Time specified by this Time_Horizon for the 'dates.start' of the Problem. The default, 0, will not exclude any OVERLOAD Problem.

Default: 0

OVERSIZE -- *a category extension of model Problem_Set*

An OVERSIZE Problem_Set selects OVERSIZE Problems with certain characteristics.

resource_plan_filter -- *a Expression field of model OVERSIZE*

An Expression that is passed a Resource_Plan as '#' and should return "true" if the OVERSIZE Problems on that Resource should be included in the Problem_Set. If it returns "false", then Problems on that Resource will be excluded.

Default: true

problem_filter -- *a Expression field of model OVERSIZE*

An Expression that is passed a Problem as '#' and should return "true" if the identified OVERSIZE Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

min_oversize -- *a Percentage field of model OVERSIZE*

A Problem will be excluded if its 'oversize' is less than the minimum Percentage specified by this Percentage_Horizon for the 'dates.start' of the Problem. The default, 0, will not exclude any OVERSIZE Problem.

Default: 0

BUCKET_OVERSIZE -- *a category extension of model Problem_Set*

A BUCKET_OVERSIZE Problem_Set selects BUCKET_OVERSIZE Problems with certain characteristics.

resource_plan_filter -- *a Expression field of model BUCKET_OVERSIZE*

An Expression that is passed a Resource_Plan as '#' and should return "true" if the BUCKET_OVERSIZE Problems on that Resource should be included in the Problem_Set. If it returns "false", then Problems on that Resource will be excluded.

Default: true

min_bucket_oversize -- *a Percentage field of model BUCKET_OVERSIZE*

A Problem will be excluded if its 'bucket_oversize' is less than the 'min_bucket_oversize' The default, 0, will not exclude any BUCKET_OVERSIZE Problem.

Default: 0

UNDERLOAD -- *a category extension of model Problem_Set*

An UNDERLOAD Problem_Set select UNDERLOAD Problems with certain characteristics.

resource_plan_filter -- *a Expression field of model UNDERLOAD*

An Expression that is passed a Resource_Plan as '#' and should return "true" if the Underload Problems on that Resource should be included in the Problem_Set. If it returns "false", then Problems on that Resource will be excluded.

Default: true

min_underload -- *a Percentage field of model UNDERLOAD*

A Problem will be excluded if its 'underload' is less than the 'min_underload' The default, 0, will not exclude any UNDERLOAD Problem.

Default: 0

RESOURCE -- *a category extension of model Problem_Set*

A RESOURCE Problem_Set selects Resource Problems (e.g. OVERLOAD, OVERSIZE, OVERTIME) on certain Resources.

Extensions	NEGATIVE_ON_HAND Extension
------------	----------------------------

resource_plan_filter -- *a Expression field of model RESOURCE*
 An Expression that is passed a Resource_Plan as '#' and should return "true" if the Problems on that Resource should be included in the Problem_Set. If it returns "false", then Problems on that Resource will be excluded.
 Default: true

problem_filter -- *a Expression field of model RESOURCE*
 An Expression that is passed a Problem as '#' and should return "true" if the identified RESOURCE Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

NEGATIVE_ON_HAND -- a category extension of model Problem_Set

A NEGATIVE_ON_HAND Problem_Set selects LOW_ON_HAND Problems with certain characteristics.

buffer_plan_filter -- *a Expression field of model NEGATIVE_ON_HAND*
 An Expression that is passed a Buffer_Plan as '#' and should return "true" if the NEGATIVE_ON_HAND Problems on that Buffer should be included in the Problem_Set. If it returns "false", then Problems on that Buffer will be excluded.
 Default: true

problem_filter -- *a Expression field of model NEGATIVE_ON_HAND*
 An Expression that is passed a Problem as '#' and should return "true" if the identified NEGATIVE_ON_HAND Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

min_deficit -- *a Quantity field of model NEGATIVE_ON_HAND*
 A Problem will be excluded if its 'deficit' is less than the minimum Quantity specified by this Quantity_Horizon for the 'dates.start' of the Problem. The default, 0, will not exclude any NEGATIVE_ON_HAND Problem.
 Default: 0
 Properties: command=True

OVER_FLOW_LIMIT -- a category extension of model Problem_Set

An OVER_FLOW_LIMIT Problem_Set selects OVER_FLOW_LIMIT Problems with certain characteristics.

Extensions	NEGATIVE_ON_HAND_AT_END Extension
------------	-----------------------------------

buffer_plan_filter -- *a Expression field of model OVER_FLOW_LIMIT*
 An Expression that is passed a Buffer_Plan as '#' and should return "true" if the OVER_FLOW_LIMIT Problems on that Buffer should be included in the Problem_Set. If it returns "false", then Problems on that Buffer will be excluded.
 Default: true

problem_filter -- *a Expression field of model OVER_FLOW_LIMIT*
 An Expression that is passed a Problem as '#' and should return "true" if the identified OVER_FLOW_LIMIT Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

min_deficit -- a Quantity field of model OVER_FLOW_LIMIT

A Problem will be excluded if its 'deficit' is less than the minimum Quantity specified by this Quantity_Horizon for the 'dates.start' of the Problem. The default, 0, will not exclude any NEGATIVE_ON_HAND Problem.

Default: 0
 Properties: command=True

NEGATIVE_ON_HAND_AT_END -- a category extension of model Problem_Set

A NEGATIVE_ON_HAND Problem_Set selects LOW_ON_HAND Problems with certain characteristics.

buffer_plan_filter -- *a Expression field of model*
NEGATIVE_ON_HAND_AT_END
 An Expression that is passed a Buffer_Plan as '#' and should return "true" if the NEGATIVE_ON_HAND Problems on that Buffer should be included in the Problem_Set. If it returns "false", then Problems on that Buffer will be excluded.
 Default: true

problem_filter -- *a Expression field of model NEGATIVE_ON_HAND_AT_END*
 An Expression that is passed a Problem as '#' and should return "true" if the identified NEGATIVE_ON_HAND_AT_END Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

min_deficit -- *a Quantity field of model NEGATIVE_ON_HAND_AT_END*
 A Problem will be excluded if its 'deficit' is less than the minimum Quantity specified by this Quantity_Horizon for the 'dates.start' of the Problem. The default, 0, will not exclude any NEGATIVE_ON_HAND Problem.
 Default: 0

Extensions	LOT_OVER_CONSUMED Extension
------------	-----------------------------

Properties: command=True

LOT_OVER_CONSUMED -- a category extension of model Problem_Set

A LOT_OVER_CONSUMED Problem_Set selects LOT_OVER_CONSUMED Problems with certain characteristics.

buffer_plan_filter -- a Expression field of model LOT_OVER_CONSUMED

An Expression that is passed a Buffer_Plan as '#' and should return "true" if the LOT_OVER_CONSUMED Problems on that Buffer should be included in the Problem_Set. If it returns "false", then Problems on that Buffer will be excluded.

Default: true

problem_filter -- a Expression field of model LOT_OVER_CONSUMED

An Expression that is passed a Problem as '#' and should return "true" if the identified LOT_OVER_CONSUMED Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

LOT_NOT_CONSUMED -- a category extension of model Problem_Set

A LOT_NOT_CONSUMED Problem_Set selects LOT_NOT_CONSUMED Problems with certain characteristics.

buffer_plan_filter -- a Expression field of model LOT_NOT_CONSUMED

An Expression that is passed a Buffer_Plan as '#' and should return "true" if the LOT_OVER_CONSUMED Problems on that Buffer should be included in the Problem_Set. If it returns "false", then Problems on that Buffer will be excluded.

Default: true

problem_filter -- a Expression field of model LOT_NOT_CONSUMED

An Expression that is passed a Problem as '#' and should return "true" if the identified LOT_NOT_CONSUMED Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

LOT_NOT_PRODUCED -- a category extension of model Problem_Set

A LOT_NOT_PRODUCED Problem_Set selects LOT_NOT_PRODUCED Problems with certain characteristics.

Extensions	LOT_OVER_PRODUCED Extension
------------	-----------------------------

buffer_plan_filter -- a Expression field of model LOT_NOT_PRODUCED

An Expression that is passed a Buffer_Plan as '#' and should return "true" if the LOT_OVER_PRODUCED Problems on that Buffer should be included in the Problem_Set. If it returns "false", then Problems on that Buffer will be excluded.

Default: true

problem_filter -- a Expression field of model LOT_NOT_PRODUCED

An Expression that is passed a Problem as '#' and should return "true" if the identified LOT_NOT_PRODUCED Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

LOT_OVER_PRODUCED -- a category extension of model Problem_Set

A LOT_OVER_PRODUCED Problem_Set selects LOT_OVER_PRODUCED Problems with certain characteristics.

buffer_plan_filter -- a Expression field of model LOT_OVER_PRODUCED

An Expression that is passed a Buffer_Plan as '#' and should return "true" if the LOT_OVER_PRODUCED Problems on that Buffer should be included in the Problem_Set. If it returns "false", then Problems on that Buffer will be excluded.

Default: true

problem_filter -- a Expression field of model LOT_OVER_PRODUCED

An Expression that is passed a Problem as '#' and should return "true" if the identified LOT_OVER_PRODUCED Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

min_excess_time -- a Time field of model LOT_OVER_PRODUCED

Problems with less excess time than min_excess_time between supplier completion date and consumer start date are excluded. This field provides a filter that can be used to allow strategy ignore small allowable excess times between consumer and supplier.

Default: 0

LOW_ON_HAND -- a category extension of model Problem_Set

A LOW_ON_HAND Problem_Set selects LOW_ON_HAND Problems with certain characteristics.

<i>Extensions</i>	<i>EXCESS_ON_HAND Extension</i>
-------------------	---------------------------------

buffer_plan_filter -- *a Expression field of model LOW_ON_HAND*
 An Expression that is passed a Buffer_Plan as '#' and should return "true" if the LOW_ON_HAND Problems on that Buffer should be included in the Problem_Set. If it returns "false", then Problems on that Buffer will be excluded.
 Default: true

problem_filter -- *a Expression field of model LOW_ON_HAND*
 An Expression that is passed a Problem as '#' and should return "true" if the identified LOW_ON_HAND Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

min_deficit -- *a Quantity field of model LOW_ON_HAND*
 A Problem will be excluded if its 'deficit' is less than the minimum Quantity specified by this Quantity_Horizon for the 'dates.start' of the Problem. The default, 0, will not exclude any LOW_ON_HAND Problem.
 Default: 0

EXCESS_ON_HAND -- *a category extension of model Problem_Set*
 An EXCESS_ON_HAND Problem_Set selects EXCESS_ON_HAND Problems with certain characteristics.

buffer_plan_filter -- *a Expression field of model EXCESS_ON_HAND*
 An Expression that is passed a Buffer_Plan as '#' and should return "true" if the EXCESS_ON_HAND Problems on that Buffer should be included in the Problem_Set. If it returns "false", then Problems on that Buffer will be excluded.
 Default: true

problem_filter -- *a Expression field of model EXCESS_ON_HAND*
 An Expression that is passed a Problem as '#' and should return "true" if the identified EXCESS_ON_HAND Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

min_excess -- *a Quantity field of model EXCESS_ON_HAND*
 A Problem will be excluded if its 'excess' is less than the minimum Quantity specified by this Quantity_Horizon for the 'dates.start' of the Problem. The default, 0, will not exclude any EXCESS_ON_HAND Problem.
 Default: 0

<i>Extensions</i>	<i>EXCESS_ON_HAND_AT_END Extension</i>
-------------------	--

EXCESS_ON_HAND_AT_END -- *a category extension of model Problem_Set*
 An EXCESS_ON_HAND_AT_END Problem_Set selects EXCESS_ON_HAND_AT_END Problems with certain characteristics.

buffer_plan_filter -- *a Expression field of model EXCESS_ON_HAND_AT_END*
 An Expression that is passed a Buffer_Plan as '#' and should return "true" if the EXCESS_ON_HAND_AT_END Problems on that Buffer should be included in the Problem_Set. If it returns "false", then Problems on that Buffer will be excluded.
 Default: true

problem_filter -- *a Expression field of model EXCESS_ON_HAND_AT_END*
 An Expression that is passed a Problem as '#' and should return "true" if the identified EXCESS_ON_HAND_AT_END Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

min_excess -- *a Quantity field of model EXCESS_ON_HAND_AT_END*
 A Problem will be excluded if its 'excess' is less than the minimum Quantity specified by this Quantity_Horizon for the 'dates.start' of the Problem. The default, 0, will not exclude any EXCESS_ON_HAND_AT_END Problem.
 Default: 0

BUFFER -- *a category extension of model Problem_Set*
 A BUFFER Problem_Set selects Buffer Problems (e.g., LOW_ON_HAND, EXCESS_ON_HAND, etc.) on certain Buffers.

buffer_plan_filter -- *a Expression field of model BUFFER*
 An Expression that is passed a Buffer_Plan as '#' and should return "true" if the Problems on that Buffer should be included in the Problem_Set. If it returns "false", then Problems on that Buffer will be excluded.
 Default: true

problem_filter -- *a Expression field of model BUFFER*
 An Expression that is passed a Problem as '#' and should return "true" if the identified BUFFER Problems should be included in the Problem_Set. If it returns "false", then the identified Problems will be excluded.
 Default: true

<i>Extensions</i>	<i>Strategy_Change Extensions</i>
-------------------	-----------------------------------

10.38 Strategy_Change Extensions

10.38.1 change_category extensions of model Strategy_Change

MOVE_IN -- a change_category extension of model Strategy_Change

MOVE_IN strategy change

MOVE_OUT -- a change_category extension of model Strategy_Change

MOVE_OUT strategy change

SPLIT -- a change_category extension of model Strategy_Change

Split strategy change

USE_MORE -- a change_category extension of model Strategy_Change

Use_More strategy change

USE_LESS -- a change_category extension of model Strategy_Change

Use_Less strategy change

USE_ALTERNATE_OPERATION -- a change_category extension of model Strategy_Change

Use_Alternate_Operation strategy change

min_alt -- a Integer field of model USE_ALTERNATE_OPERATION
The min_alt and max_alt defines the range of alternate operations considered. For example, if min_alt = max_alt = 0, only the first alternate is allowed.
Default: 0

max_alt -- a Integer field of model USE_ALTERNATE_OPERATION
See notes in min_alt.
Default: INT_MAX

USE_ALTERNATE_RESOURCE -- a change_category extension of model Strategy_Change

Use_Alternate_Resource strategy change

<i>Extensions</i>	<i>USE_EFFECTIVE_ALTERNATE Extension</i>
-------------------	--

USE_EFFECTIVE_ALTERNATE -- a change_category extension of model Strategy_Change

Use_Effective_Alternate strategy change

INCREASE_CAPACITY -- a change_category extension of model Strategy_Change

Increase_Capacity strategy change

DECREASE_CAPACITY -- a change_category extension of model Strategy_Change

Decrease_Capacity strategy change

RESIZE_MORE -- a change_category extension of model Strategy_Change

Resize_More strategy change

RESIZE_LESS -- a change_category extension of model Strategy_Change

Resize_Less strategy change

UPSTREAM -- a change_category extension of model Strategy_Change

Upstream strategy change

DOWNSTREAM -- a change_category extension of model Strategy_Change

Downstream strategy change

<i>Extensions</i>	<i>Strategy_Lock Extensions</i>
-------------------	---------------------------------

10.39 Strategy_Lock Extensions

10.39.1 spec extensions of model Strategy_Lock

OPERATION_PLAN_RANK_RANGE -- *a spec extension of model Strategy_Lock*

The OPERATION_PLAN_RANK_RANGE 'spec' extension allows users to specify a range which will lock Operation_Plan's whose rank 'lies' within it.

min_rank -- *a Number field of model OPERATION_PLAN_RANK_RANGE*
 Specifies the min of the range, within which Operation_Plan's will be locked.
 Default: -oo

max_rank -- *a Number field of model OPERATION_PLAN_RANK_RANGE*
 Specifies the max of the range, within which Operation_Plan's will be locked.
 Default: oo

OPERATION_PLAN_RANK_EXPRESSION -- *a spec extension of model Strategy_Lock*

The OPERATION_PLAN_EXPRESSION 'spec' extension specifies an Expression that is passed an Operation_Plan as 'w' and should return "true" if the given Operation_Plan should be considered locked.

expression -- *a Expression field of model OPERATION_PLAN_RANK_EXPRESSION*
 An Expression that is passed an Operation_Plan as 'w' and that Operation_Plan will be locked if it returns "true".
 Default: false

<i>Extensions</i>	<i>Strategy_Goal Extensions</i>
-------------------	---------------------------------

10.40 Strategy_Goal Extensions

10.40.1 goal extensions of model Strategy_Goal

FEASIBILITY -- *a goal extension of model Strategy_Goal*

The goal is to minimize the sum of 'interaction' of infeasible Problems, ultimately driving it to zero by eliminating all infeasible Problems.

target_interaction -- *a Number field of model FEASIBILITY*
 The target level for the sum of 'interaction' of all infeasible Problems. If that sum is less than or equal to target_interaction, then the target has been met (though the goal to minimize it to zero still remains).

This is typically used in conjunction with target-oriented 'termination' extensions.
 Default: 0

MINIMIZE_PROBLEM_COUNT -- *a goal extension of model Strategy_Goal*

The goal is to minimize the number of Problems, ultimately driving it to zero by eliminating all Problems. Most applications should use MINIMIZE_PROBLEMS instead of MINIMIZE_PROBLEM_COUNT.

target_problem_count -- *a Number field of model MINIMIZE_PROBLEM_COUNT*
 The target level for the number of Problems. If that number is less than or equal to target_problem_count, then the target has been met (though the goal to minimize it to zero still remains).

This is typically used in conjunction with target-oriented 'termination' extensions.
 Default: 0

MINIMIZE_PROBLEMS -- *a goal extension of model Strategy_Goal*

The goal is to minimize problems in all problem sets. This goal differs from MINIMIZE_PROBLEM_COUNT in that problems are internally assigned a weight based on their size and importance, and the strategy tries to minimize the sum of these weights (instead of the number of problems).

Extensions**MINIMIZE_LATENESS Extension**

target_problems -- *a Number field of model MINIMIZE_PROBLEMS*

The target level for the sum of the weights of all problems. If that number is less than or equal to **target_problems**, then the target has been met (though the goal to minimize it to zero still remains).

This is typically used in conjunction with target-oriented termination extensions.

Default: 0

MINIMIZE_LATENESS -- a goal extension of model Strategy_Goal

The goal is to minimize the sum of 'lateness' of all item requests.

target_lateness -- *a Number field of model MINIMIZE_LATENESS*

The target level for the sum of 'lateness' of all requests. If that sum is less than or equal to **target_lateness**, then the target has been met (though the goal to minimize it to zero still remains).

This is typically used in conjunction with target-oriented termination extensions.

Default: 0

WEIGHTED_LATENESS -- a goal extension of model Strategy_Goal

The goal is to minimize the weighted sum of 'lateness' of all item requests.

target_lateness -- *a Number field of model WEIGHTED_LATENESS*

The target level for the sum of 'lateness' of all requests. If that sum is less than or equal to **target_lateness**, then the target has been met (though the goal to minimize it to zero still remains).

This is typically used in conjunction with target-oriented termination extensions.

Default: 0

day_lat_power -- *a Number field of model WEIGHTED_LATENESS*

The weight for late order. Weighted lateness equals days late raised to **day_lat_power** times quantity late raised to **quantity_lat_power**.

Default: 1.0

quantity_lat_power -- *a Number field of model WEIGHTED_LATENESS*

The weight for late order. Weighted lateness equals days late raised to **day_lat_power** times quantity late raised to **quantity_lat_power**.

Default: 0.0

Extensions**MINIMIZE_SHORTNESS Extension**

MINIMIZE_SHORTNESS -- *a goal extension of model Strategy_Goal*

The goal is to minimize the sum of 'shortness' of all item requests.

target_shortness -- *a Number field of model MINIMIZE_SHORTNESS*

The target level for the sum of 'shortness' of all requests. If that sum is less than or equal to **target_shortness**, then the target has been met (though the goal to minimize it to zero still remains).

This is typically used in conjunction with target-oriented termination extensions.

Default: 0

WEIGHTED_SHORTNESS -- a goal extension of model Strategy_Goal

The goal is to minimize the weighted sum of 'shortness' of all item requests.

target_shortness -- *a Number field of model WEIGHTED_SHORTNESS*

The target level for the sum of 'shortness' of all requests. If that sum is less than or equal to **target_lateness**, then the target has been met (though the goal to minimize it to zero still remains).

This is typically used in conjunction with target-oriented termination extensions.

Default: 0

quantity_short_power -- *a Number field of model WEIGHTED_SHORTNESS*

The weight for short order. Weighted shortness equals quantity short raised to **quantity_short_power**.

Default: 1.0

MINIMIZE_COST -- a goal extension of model Strategy_Goal

The goal is to minimize the total 'cost' of the plan.

target_cost -- *a Number field of model MINIMIZE_COST*

The target level for 'cost'. If 'cost' is less than or equal to **target_cost**, then the target has been met (though the goal to minimize cost to zero still remains). This is typically used in conjunction with target-oriented termination extensions.

Default: 0

MAXIMIZE_PROFIT -- a goal extension of model Strategy_Goal

Extensions	MAXIMIZE_REVENUE Extension
------------	----------------------------

The goal is to maximize the difference between 'revenue' and 'cost'. Revenue is currently taken to be the sum of the planned price of the item requests in the committed forecast that are satisfied by the current plan.

target_profit -- *a Number field of model MAXIMIZE_PROFIT*

The target level for 'profit'. If 'profit' is greater than or equal to target_profit, then the target has been met (though the goal to maximize it still remains).

This is typically used in conjunction with target-oriented termination extensions.

Default: 0

MAXIMIZE_REVENUE -- *a goal extension of model Strategy_Goal*

The goal is to maximize 'revenue'. Revenue is currently taken to be the sum of the planned price of the item requests in the committed forecast that are satisfied by the current plan.

target_revenue -- *a Number field of model MAXIMIZE_REVENUE*

The target level for 'revenue'. If 'revenue' is greater than or equal to target_revenue, then the target has been met (though the goal to maximize it still remains).

This is typically used in conjunction with target-oriented termination extensions.

Default: 0

Extensions	Calendar_Entry Extensions
------------	---------------------------

10.41 Calendar_Entry Extensions

10.41.1 value extensions of model Calendar_Entry

NUMBER -- *a value extension of model Calendar_Entry*

Entry specifies a single Number.

number -- *a Number field of model NUMBER*

Number associated with this entry

Default: 0

QUANTITY -- *a value extension of model Calendar_Entry*

Entry specifies a single Quantity.

quantity -- *a Quantity field of model QUANTITY*

Quantity associated with this entry

Default: 0 (unitless)

NUMBER_QUANTITY -- *a value extension of model Calendar_Entry*

Entry specifies a Number and a Quantity.

number -- *a Number field of model NUMBER_QUANTITY*

Number associated with this entry

Default: 0

quantity -- *a Quantity field of model NUMBER_QUANTITY*

Quantity associated with this entry

Default: 0 (unitless)

SYMBOL -- *a value extension of model Calendar_Entry*

Entry specifies a single Symbol.

symbol -- *a Symbol field of model SYMBOL*

Symbol associated with this entry

Default: none

TIME -- *a value extension of model Calendar_Entry*

Entry specifies a single Time

Extensions	day_pattern extensions of model Calendar_Entry
------------	--

time -- a Time field of model TIME
Time associated with this entry
Default: 0

10.41.2 day_pattern extensions of model Calendar_Entry

EVERYDAY -- a day_pattern extension of model Calendar_Entry

An EVERYDAY Calendar_Entry extension is applicable to every single day, or Date, within the Calendar_Entry's effective range.

EVERY_N_DAYS -- a day_pattern extension of model Calendar_Entry

An EVERY_N_DAYS Calendar_Entry extension is applicable to every 'n'th day, or Date, within the Calendar_Entry's effective range, where the effective_start Date is the first applicable day. For instance, every 15 days between Jan. 1, 1996 and Jul. 31, 1996. If effective_start is "oo_past", or "oo_future", then the Calendar_Entry does not apply to any day.

nth -- a Integer field of model EVERY_N_DAYS

The interval between any applicable day and the next applicable day. If 'nth' is 15, and effective_start is "Jan 1", then the applicable days are: Jan. 1, Jan. 16, Jan. 31, Feb. 15, ... The default is 1, which is equivalent to the EVERYDAY Calendar_Entry extension.
Default: 1

WEEKDAYS -- a day_pattern extension of model Calendar_Entry

A WEEKDAYS Calendar_Entry extension applies to all weekdays (Monday through Friday) in the effective range of the Calendar_Entry. See the WEEKENDS extension for the inverse.

WEEKENDS -- a day_pattern extension of model Calendar_Entry

A WEEKENDS Calendar_Entry extension applies to all weekends (Saturday and Sunday) in the effective range of the Calendar_Entry. See the WEEKDAYS extension for the inverse.

DAYS_OF_WEEK -- a day_pattern extension of model Calendar_Entry

Extensions	DAYS_OF_WEEK Extension
------------	------------------------

A DAYS_OF_WEEK Calendar_Entry extension applies to the specified days of the week that are in the Calendar_Entry's effective range. For instance, every Monday, Wednesday, and Friday between Jan. 1, 1996 and July 31, 1996.

days -- a String field of model DAYS_OF_WEEK

A list that specifies the applicable days of the week for this Calendar_Entry. Entries in the list are separated by commas or white space. Each element of the list may be a single day (e.g., Mon), or a range of days (e.g., Mon-Fri). The following are valid: "Mo,Tu,We", "Sa,Su", "Monday-Friday", "Tu,Th", or "Mo-Tu,Th,Fri,Sa". Note that "Mo-Fri" is equivalent to the WEEKDAYS extension, and "Sa,Su" is equivalent to the WEEKENDS extension. If no days are specified, the default is "Mo-Su" which is equivalent to the EVERYDAY extension.

Because the names of the week days change in internationalized versions, be sure to specify enough characters of each day to be non-ambiguous. For example, in English, "M-F" is not ambiguous, while "M,T,W,T,F" is ambiguous. The latter should be specified as "M,Tu,W,Th,F" to avoid any ambiguity.
Default: Mo-Su

DAYS_OF_MONTH -- a day_pattern extension of model Calendar_Entry

A DAYS_OF_MONTH Calendar_Entry extension applies to a specific day in a month, a subset of months, or all months. For example, the 15th day of each month, the 25th day of December, the first day of each quarter (Jan, Apr, Jul, Oct), the 1-th day of each month (the last day of each month), and so on, are examples.

day -- a Integer field of model DAYS_OF_MONTH

The day of month on which this Calendar_Entry is applicable. Valid values are [1-31] inclusive which represent the 1st, 2nd, 3rd, etc. days of the month. The default is the first day of the month (1).
Default: 1

months -- a String field of model DAYS_OF_MONTH

A list that specifies the months in which this Calendar_Entry is applicable. Entries in the list may be separated by commas or white space. Each element of the list may be a single month (e.g., January), a range of months (e.g., May-Aug). The following are valid: "J,F,Mar", "Jun-Jul", "May-Aug" or "Jan-Mar,Jun-Nov,Dec". At least the first 3 characters of the name of each month must be specified. The default is "Jan-Dec" which specifies all months.

Extensions	DAY_OF_WEEK_OF_MONTH Extension
------------	--------------------------------

Because the names of the months change in internationalized versions, be sure to specify enough characters of each month to be non-ambiguous. For example, in English, "J-D" is not ambiguous, while "J.F.M.A.M.J" is ambiguous. The latter should be specified as "J.F.Mar.Ap.May.Jun" to avoid any ambiguity.
Default: Jan-Dec

DAY_OF_WEEK_OF_MONTH -- a day_pattern extension of model Calendar_Entry

A DAY_OF_WEEK_OF_MONTH Calendar_Entry pattern applies to the nth occurrence of a specific day of week of a month, a subset of months, or all months. For example, the 4th Thursday in November, the 1st Tuesday in Oct-Dec, the 1st Monday in September, or the 3rd Wednesday of each month.

day -- a Integer field of model DAY_OF_WEEK_OF_MONTH

The day of the week on which this Calendar_Entry is applicable. Valid values are [1-7] where the value of 1 is Monday, 2 is Tuesday, ..., and 7 is Sunday. The default is Monday.

Default: 1

nth -- a Integer field of model DAY_OF_WEEK_OF_MONTH

Specifies the nth occurrence of the day of week on which this Calendar_Entry is applicable. Valid values are [1-5] where the value of 1 is the 1st occurrence, 2 is the 2nd occurrence, etc. For example, the 4th Thursday or the 1st Friday. The default is the first occurrence.

Default: 1

week -- a Integer field of model DAY_OF_WEEK_OF_MONTH

This field is obsolete and will be removed in 3.06_BETA Use 'nhb' instead. Refer to 'nhb' for documentation.

Default: 1

Properties: obsolete=True

months -- a String field of model DAY_OF_WEEK_OF_MONTH

A list that specifies the months in which this Calendar_Entry is applicable. Entries in the list may be separated by commas or white space. Each element of the list may be a single month (e.g., January), a range of months (e.g., May-Aug). The following are valid: "J.J.Mar", "Jun Jul", "May-Aug" or "Jan-Mar,Jun-Nov-Dec". At least the first 3 characters of the name of each month must be specified. The default is "Jan-Dec" which specifies all months.

Extensions	DAY_OF_LAST_WEEK_OF_MONTH Extension
------------	-------------------------------------

Because the names of the months change in internationalized versions, be sure to specify enough characters of each month to be non-ambiguous. For example, in English, "J-D" is not ambiguous, while "J.F.M.A.M.J" is ambiguous. The latter should be specified as "J.F.Mar.Ap.May.Jun" to avoid any ambiguity.
Default: Jan-Dec

DAY_OF_LAST_WEEK_OF_MONTH -- a day_pattern extension of model Calendar_Entry

A DAY_OF_LAST_WEEK_OF_MONTH Calendar_Entry pattern applies to a specific day in the last week of a month, a subset of months, or all months. For example, the last Monday of May or the last Friday of each month.

day -- a Integer field of model DAY_OF_LAST_WEEK_OF_MONTH

The day of the week on which this Calendar_Entry is applicable. Valid values are [1-7] where the value of 1 is Monday, 2 is Tuesday, ..., and 7 is Sunday. The default is Monday.

Default: 1

months -- a String field of model DAY_OF_LAST_WEEK_OF_MONTH

A list that specifies the months in which this Calendar_Entry is applicable. Entries in the list may be separated by commas or white space. Each element of the list may be a single month (e.g., January), a range of months (e.g., May-Aug). The following are valid: "J.J.Mar", "Jun Jul", "May-Aug" or "Jan-Mar,Jun-Nov-Dec". At least the first 3 characters of the name of each month must be specified. The default is "Jan-Dec" which specifies all months.

Because the names of the months change in internationalized versions, be sure to specify enough characters of each month to be non-ambiguous. For example, in English, "J-D" is not ambiguous, while "J.F.M.A.M.J" is ambiguous. The latter should be specified as "J.F.Mar.Ap.May.Jun" to avoid any ambiguity.
Default: Jan-Dec

DAY_OF_YEAR -- a day_pattern extension of model Calendar_Entry

A DAY_OF_YEAR day_pattern Calendar_Entry applies to the Nth day of each year. For instance the 183rd day of the year specifies the mid-point of the year.

day -- a Integer field of model DAY_OF_YEAR

The day of the year on which this Calendar_Entry is applicable. Valid values are [1-366] inclusive where the value of 1 is Jan 1. The default is Jan 1.
Default: 1

Extensions	YEARLY Extension
------------	------------------

YEARLY -- a day_pattern extension of model Calendar_Entry

A YEARLY day_pattern Calendar_Entry applies to a specific day in a specific month of each year. For instance, the 25th of December, the 4th of July, and the first of January.

month -- a Integer field of model YEARLY

Specifies the month of year in which this Calendar_Entry is applicable. Valid values are [1-12] inclusive, where the value of 1 is January, 2 is February, ... and 12 is December. The default is January.

Default: 1

day -- a Integer field of model YEARLY

Specifies the day of month on which this Calendar_Entry is applicable. Valid values are [1-31] which represent the 1st, 2nd, 3rd, etc. days of the month. The default is the first day of the month (1).

Default: 1

Extensions	Calendar Extensions
------------	---------------------

10.42 Calendar Extensions

10.42.1 entry_value extensions of model Calendar

UNSPECIFIED -- a entry_value extension of model Calendar

Calendar's entry_value is not specified

NUMBER -- a entry_value extension of model Calendar

Calendar's entries specify a single Number.

default_number -- a Number field of model NUMBER

Default Number value for calendar This will be the value of this Calendar at any time in which no value was specified by the Calendar Entries. The default value of default_number is 0.

Default: 0

QUANTITY -- a entry_value extension of model Calendar

Calendar's entries specify a single Number.

default_quantity -- a Quantity field of model QUANTITY

Default Quantity value for calendar This will be the value of this Calendar at any time in which no value was specified by the Calendar Entries. The default value of default_quantity is 0 (unitless)

Default: 0 (unitless)

NUMBER_QUANTITY -- a entry_value extension of model Calendar

Calendar's entries specify a single Number_Quantity_Pair

default_number -- a Number field of model NUMBER_QUANTITY

Default Number value of NUMBER_QUANTITY This will be the value of this Calendar at any time in which no value was specified by the Calendar Entries. The default value of default_number is 0.

Default: 0

default_quantity -- a Quantity field of model NUMBER_QUANTITY

Default Quantity value of NUMBER_QUANTITY This will be the value of this Calendar at any time in which no value was specified by the Calendar Entries. The default value of default_quantity is 0 (unitless)

Default: 0 (unitless)

<i>Extensions</i>	<i>SYMBOL Extension</i>
-------------------	-------------------------

SYMBOL -- *a entry_value extension of model Calendar*

Calendar's entries specify a single Symbol

default_value -- *a Symbol field of model SYMBOL*

Default Symbol value for calendar This will be the value of this Calendar at any time in which no value was specified by the Calendar Entries. There is no default value of this default_value.

Default: none

TIME -- *a entry_value extension of model Calendar*

Calendar's entries specify a single Number.

default_time -- *a Time field of model TIME*

Default Time value for calendar This will be the value of this Calendar at any time in which no value was specified by the Calendar Entries. The default value of default_time is 0.0.

Default: 0.0

<i>Extensions</i>	<i>Flow_Criterion Extensions</i>
-------------------	----------------------------------

10.43 Flow_Criterion Extensions

10.43.1 criterion extensions of model Flow_Criterion

CUSTOMER_RANK -- *a criterion extension of model Flow_Criterion*

A CUSTOMER_RANK Flow_Criterion uses the 'rank' field of the customer Site (delivery_request.customer_plan.site.rank) as a relative weighting or sort criterion.

produce_separately -- *a Logical field of model CUSTOMER_RANK*

If 'true', then when generating producing Operation_Plan, it will generate a separate Operation_Plan in each bucket for each different customer rank of consuming Flow_Plan.

Default: false

default_rank -- *a Number field of model CUSTOMER_RANK*

If there is no customer (or no Delivery_Request) associated with the Flow_Plan, then this default_rank is used instead of the customer rank.

Default: 0

SELLER_RANK -- *a criterion extension of model Flow_Criterion*

A SELLER_RANK Flow_Criterion uses the 'rank' field of the Seller as a relative weighting or sort criterion.

produce_separately -- *a Logical field of model SELLER_RANK*

If 'true', then when generating producing Operation_Plan, it will generate a separate Operation_Plan in each bucket for each different Seller rank of consuming Flow_Plan.

Default: false

default_rank -- *a Number field of model SELLER_RANK*

If there is no Seller (or no Delivery_Request) associated with the Flow_Plan, then this default_rank is used instead of the customer rank.

Default: 0

REQUEST_RANK -- *a criterion extension of model Flow_Criterion*

A REQUEST_RANK Flow_Criterion uses the 'rank' field of the Delivery_Request, as ranked by its 'customer_plan' as a relative weighting or sort criterion.

<i>Extensions</i>	<i>ACTUAL_OR_FORECAST Extension</i>
-------------------	-------------------------------------

produce_separately -- *a Logical field of model REQUEST_RANK*
 If "true", then when generating producing Operation_Plan, it will generate a separate Operation_Plan in each bucket for each different Delivery_Request rank of consuming Flow_Plan.
 Default: false

default_rank -- *a Number field of model REQUEST_RANK*
 If there is no Delivery_Request associated with the Flow_Plan, then this 'default_rank' is used instead of the customer rank.
 Default: 0

ACTUAL_OR_FORECAST -- *a criterion extension of model Flow_Criterion*

A ACTUAL_OR_FORECAST Flow_Criterion uses the Number 1 if the Flow_Plan is associated with an actual Delivery_Request, the Number 0 if the Flow_Plan is associated with a forecast Delivery_Request, or the 'default_rank' if the Flow_Plan is not associated with a Delivery_Request, as a relative weighting or sort criterion.

produce_separately -- *a Logical field of model ACTUAL_OR_FORECAST*
 If "true", then when generating producing Operation_Plan, it will generate a separate Operation_Plan in each bucket for actuals, forecasts, and other consuming Flow_Plan.
 Default: false

default_rank -- *a Number field of model ACTUAL_OR_FORECAST*
 If there is no Delivery_Request associated with the Flow_Plan, then this 'default_rank' is used instead of 1 or 0.
 Default: 0

REQUEST_ISSUED -- *a criterion extension of model Flow_Criterion*

A REQUEST_ISSUED Flow_Criterion uses the 'issued' field of the Request as a relative weighting or sort criterion.

produce_separately -- *a Logical field of model REQUEST_ISSUED*
 If "true", then when generating producing Operation_Plan, it will generate a separate Operation_Plan in each bucket for each different customer rank of consuming Flow_Plan.
 Default: false

<i>Extensions</i>	<i>PROMISE_DUE Extension</i>
-------------------	------------------------------

default_rank -- *a Number field of model REQUEST_ISSUED*
 If there is no customer (or no Delivery_Request) associated with the Flow_Plan, then this 'default_rank' is used instead of the customer rank.
 Default: 0

default_issued -- *a Date field of model REQUEST_ISSUED*
 If there is no Request associated with the Flow_Plan, then this 'default_issued' Date is used instead.
 Default: 0

PROMISE_DUE -- *a criterion extension of model Flow_Criterion*

A PROMISE_DUE Flow_Criterion uses the 'due end' field of the Delivery_Promise as a relative weighting or sort criterion.

produce_separately -- *a Logical field of model PROMISE_DUE*
 If "true", then when generating producing Operation_Plan, it will generate a separate Operation_Plan in each bucket for each different customer rank of consuming Flow_Plan.
 Default: false

default_rank -- *a Number field of model PROMISE_DUE*
 If there is no customer (or no Delivery_Request) associated with the Flow_Plan, then this 'default_rank' is used instead of the customer rank.
 Default: 0

default_due -- *a Date field of model PROMISE_DUE*
 If there is no Delivery_Promise associated with the Flow_Plan, then this 'default_due' Date is used instead.
 Default: 0

REQUEST_DUE -- *a criterion extension of model Flow_Criterion*

A REQUEST_DUE Flow_Criterion uses the 'due end' field of the Delivery_Promise as a relative weighting or sort criterion.

produce_separately -- *a Logical field of model REQUEST_DUE*
 If "true", then when generating producing Operation_Plan, it will generate a separate Operation_Plan in each bucket for each different customer rank of consuming Flow_Plan.
 Default: false

<i>Extensions</i>	<i>DUE_DATE Extension</i>
-------------------	---------------------------

default_rank -- *a Number field of model REQUEST_DUE*
 If there is no customer (or no Delivery_Request) associated with the Flow_Plan, then this default_rank is used instead of the customer rank.
 Default: 0

default_due -- *a Date field of model REQUEST_DUE*
 If there is no Delivery_Promise associated with the Flow_Plan, then this 'default_due' Date is used instead.
 Default: 0

DUE_DATE -- *a criterion extension of model Flow_Criterion*

A DUE_DATE Flow_Criterion uses the accepted due date, if it exists, or the promise due date otherwise, as a relative weighing or sort criterion.

produce_separately -- *a Logical field of model DUE_DATE*
 If "true", then when generating producing Operation_Plans, it will generate a separate Operation_Plan in each bucket for each different customer rank of consuming Flow_Plan.
 Default: false

default_rank -- *a Number field of model DUE_DATE*
 If there is no customer (or no Delivery_Request) associated with the Flow_Plan, then this 'default_rank' is used instead of the customer rank.
 Default: 0

default_due -- *a Date field of model DUE_DATE*
 If there is no Delivery_Promise associated with the Flow_Plan, then this 'default_due' Date is used instead.
 Default: 0

PROMISED -- *a criterion extension of model Flow_Criterion*

A PROMISED Flow_Criterion uses the Number 1 if the Flow_Plan is associated with a Delivery_Promise with 'due end' Date in the same bucket as the 'due end' Date of the Delivery_Request, the Number 0 if they are in different buckets, or the 'default_rank' if the Flow_Plan is not associated with a Delivery_Request, as a relative weighing or sort criterion.

<i>Extensions</i>	<i>ENTRY_DATE Extension</i>
-------------------	-----------------------------

produce_separately -- *a Logical field of model PROMISED*
 If "true", then when generating producing Operation_Plans, it will generate a separate Operation_Plan in each bucket for each different ranking (1, 0, or default_rank) of consuming Flow_Plan.
 Default: false

default_rank -- *a Number field of model PROMISED*
 If there is no Delivery_Request associated with the Flow_Plan, then this 'default_rank' is used instead of 1 or 0.
 Default: 0

ENTRY_DATE -- *a criterion extension of model Flow_Criterion*

An ENTRY_DATE Flow_Criterion uses the order entry date as a relative weighing or sort criterion.

produce_separately -- *a Logical field of model ENTRY_DATE*
 If "true", then when generating producing Operation_Plans, it will generate a separate Operation_Plan in each bucket for each different customer rank of consuming Flow_Plan.
 Default: false

default_rank -- *a Number field of model ENTRY_DATE*
 If there is no customer (or no Delivery_Request) associated with the Flow_Plan, then this 'default_rank' is used instead of the customer rank.
 Default: 0

<i>Extensions</i>	<i>Calendar_Plan Extensions</i>
-------------------	---------------------------------

10.44 Calendar_Plan Extensions

10.44.1 entry_value extensions of model Calendar_Plan

NUMBER -- a entry_value extension of model Calendar_Plan

A NUMBER Calendar_Plan contains a Number for each Date. A NUMBER Calendar dictates the values during the 'horizon', and the before_horizon_number' and after_horizon_number' fields specify the values used beyond the horizon'.

before_horizon_number -- a Number field of model NUMBER
This will be the value of this Calendar_Plan at any Date before the horizon'.

This is not settable. It is dictated by the model that provides the Calendar_Plan field. See the documentation for the Calendar_Plan field of interest.

Properties: Export-Only Field

after_horizon_number -- a Number field of model NUMBER
This will be the value of this Calendar_Plan at any Date after the horizon'.

This is not settable. It is dictated by the model that provides the Calendar_Plan field. See the documentation for the Calendar_Plan field of interest.

Properties: Export-Only Field

intra_horizon_number -- a Number field of model NUMBER

This will be the value of this Calendar_Plan at any Date within the horizon' where no other Calendar_Entry is active. If not set, the default from 'calendar' is used.
Default: 0

QUANTITY -- a entry_value extension of model Calendar_Plan

A QUANTITY Calendar_Plan contains a Quantity for each Date. A QUANTITY Calendar dictates the values during the 'horizon', and the before_horizon_quantity' and after_horizon_quantity' fields specify the values used beyond the horizon'.

Note that currently only unitless Quantities are supported. In the future, the QUANTITY Calendar will have a unit' field which defines the Unit to which all the entries' quantities are converted.

before_horizon_quantity -- a Quantity field of model QUANTITY
This will be the value of this Calendar_Plan at any Date before the horizon'.

<i>Extensions</i>	<i>NUMBER_QUANTITY Extension</i>
-------------------	----------------------------------

This is not settable. It is dictated by the model that provides the Calendar_Plan field. See the documentation for the Calendar_Plan field of interest.

Properties: Export-Only Field

after_horizon_quantity -- a Quantity field of model QUANTITY
This will be the value of this Calendar_Plan at any Date after the horizon'.

This is not settable. It is dictated by the model that provides the Calendar_Plan field. See the documentation for the Calendar_Plan field of interest.

Properties: Export-Only Field

intra_horizon_quantity -- a Quantity field of model QUANTITY

This will be the value of this Calendar_Plan at any Date within the horizon' where no other Calendar_Entry is active. If not set, the default from 'calendar' is used.
Default: 0 (unitless)

NUMBER_QUANTITY -- a entry_value extension of model Calendar_Plan

A NUMBER_QUANTITY Calendar_Plan contains a Number and a Quantity for each Date. A NUMBER_QUANTITY Calendar dictates the values during the 'horizon', and the before_horizon_number', after_horizon_number', before_horizon_quantity', and after_horizon_quantity' fields specify the values used beyond the horizon'.

Note that currently only unitless Quantities are supported. In the future, the QUANTITY Calendar will have a unit' field which defines the Unit to which all the entries' quantities are converted.

before_horizon_number -- a Number field of model NUMBER_QUANTITY
This will be the value of this Calendar_Plan at any Date before the horizon'.

This is not settable. It is dictated by the model that provides the Calendar_Plan field. See the documentation for the Calendar_Plan field of interest.

Properties: Export-Only Field

after_horizon_number -- a Number field of model NUMBER_QUANTITY
This will be the value of this Calendar_Plan at any Date after the horizon'.

This is not settable. It is dictated by the model that provides the Calendar_Plan field. See the documentation for the Calendar_Plan field of interest.

Properties: Export-Only Field

<i>Extensions</i>	<i>SYMBOL Extension</i>
-------------------	-------------------------

intra_horizon_number - - *a* Number *field of model* NUMBER_QUANTITY
 This will be the value of this Calendar_Plan at any Date within the horizon' where no other Calendar_Entry is active. If not set, the default from 'calendar' is used.
 Default: 0

before_horizon_quantity - - *a* Quantity *field of model* NUMBER_QUANTITY
 This will be the value of this Calendar_Plan at any Date before the horizon'.

This is not settable. It is dictated by the model that provides the Calendar_Plan field.
 See the documentation for the Calendar_Plan field of interest.

Properties: Export-Only Field

after_horizon_quantity - - *a* Quantity *field of model* NUMBER_QUANTITY
 This will be the value of this Calendar_Plan at any Date after the horizon'.

This is not settable. It is dictated by the model that provides the Calendar_Plan field.
 See the documentation for the Calendar_Plan field of interest.

Properties: Export-Only Field

intra_horizon_quantity - - *a* Quantity *field of model* NUMBER_QUANTITY
 This will be the value of this Calendar_Plan at any Date within the horizon' where no other Calendar_Entry is active. If not set, the default from 'calendar' is used.
 Default: 0 (unitless)

SYMBOL - - *a* entry_value *extension of model* Calendar_Plan

A SYMBOL.Calendar_Plan contains a Symbol for each Date. A SYMBOL.Calendar dictates the values during the horizon', and the before_horizon_symbol' and after_horizon_symbol' fields specify the values used beyond the horizon'.

before_horizon_symbol - - *a* Symbol *field of model* SYMBOL

This will be the value of this Calendar_Plan at any Date before the horizon'.

This is not settable. It is dictated by the model that provides the Calendar_Plan field.
 See the documentation for the Calendar_Plan field of interest.

Properties: Export-Only Field

after_horizon_symbol - - *a* Symbol *field of model* SYMBOL

This will be the value of this Calendar_Plan at any Date after the horizon'.

This is not settable. It is dictated by the model that provides the Calendar_Plan field.
 See the documentation for the Calendar_Plan field of interest.

<i>Extensions</i>	<i>TIME Extension</i>
-------------------	-----------------------

Properties: Export-Only Field

intra_horizon_symbol - - *a* Symbol *field of model* SYMBOL

This will be the value of this Calendar_Plan at any Date within the horizon' where no other Calendar_Entry is active. If not set, the default from 'calendar' is used.

Default: none

TIME - - *a* entry_value *extension of model* Calendar_Plan

A TIME.Calendar_Plan contains a Time for each Date. A TIME.Calendar dictates the values during the horizon', and the before_horizon_time' and after_horizon_time' fields specify the values used beyond the horizon'.

before_horizon_time - - *a* Time *field of model* TIME
 This will be the value of this Calendar_Plan at any Date before the horizon'.

This is not settable. It is dictated by the model that provides the Calendar_Plan field.
 See the documentation for the Calendar_Plan field of interest.

Properties: Export-Only Field

after_horizon_time - - *a* Time *field of model* TIME
 This will be the value of this Calendar_Plan at any Date after the horizon'.

This is not settable. It is dictated by the model that provides the Calendar_Plan field.
 See the documentation for the Calendar_Plan field of interest.

Properties: Export-Only Field

intra_horizon_time - - *a* Time *field of model* TIME

This will be the value of this Calendar_Plan at any Date within the horizon' where no other Calendar_Entry is active. If not set, the default from 'calendar' is used.
 Default: 00:00

10.45 Format Extensions

10.45.1 spec extensions of model Format

Void -- a spec extension of model Format

A Format which can handle Void values (or lack of such values).

These models have a field that is a Void model :

Plan, Operation_Plan, Resource_Plan, Forecast, Seller_Plan, Problem, Active_Strategy, LINK, Operation_State, Request, Promise, Acceptance, Delivery_Request, Delivery_Promise, Load_Plan, Item_Acceptance, Forecast_Entry, Item_Request, Item_Promise, Item_Available_To_Promise, Product_Available_To_Promise, MEMBER_RANK, ALTERNATES_PRIMARY, ALTERNATES_PROPORTIONAL, EFFECTIVE_CALENDAR, Consolidation, BUCKETED_NESTED_SORT, User, Data_Directory.

specification -- a Symbol field of model Void

Specifies how to format void values

Default: Empty String

Logical -- a spec extension of model Format

A Format which can handle Logical values.

These models have a field that is a Logical model :

Site, Seller, Operation, Operation_Plan, Product, Forecast, Problem, Active_Strategy, LINK, Location, Item, Configuration, Item_Group, CUSTOMER, Product_Group, Delivery_Request, Delivery_Promise, Delivery_Acceptance, Strategy, Problem_Set, Active_Problem, MANUAL, Unit, Forecast_Entry, GROUP, Item_Request, Flow, Operation_Problem_Detector, Load_Plan, Flow_Plan, Item_Acceptance, Item_Promise, Buffer_Problem_Detector, Lot, Box, DELIVER, Resource_Problem_Detector, MEMBER_RANK, ALTERNATES_PRIMARY, ALTERNATES_MANUAL, ALTERNATES_PROPORTIONAL, ALTERNATES_PREFERENCE, EFFECTIVE_CALENDAR, MPPS_Operation_Resource, MPPS_BASIC, CALENDAR_RATE, BUCKETED_NESTED_SORT, Sorted_Bucket, PRODUCING_FLOW_CALENDAR, PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK,

CUSTOMER_RANK, SELLER_RANK, REQUEST_RANK, PROMISE_RANK, ACTUAL_OR_FORECAST, REQUEST_ISSUED, PROMISE_DUE, REQUEST_DUE, DUE_DATE, PROMISED, ENTRY_DATE, Feature, User, Data_Directory, Report_Directory, Format, Model_Type, Field.

specification -- a Symbol field of model Logical

Specifies how to format Logical values

Default: False, True

String -- a spec extension of model Format

A Format which can handle Strings.

These models have a field that is a String model :

Supply_Chain, Site, Seller, Plan, Problem_Category, Operation, Operation_Plan, Resource, Resource_Plan, Buffer, Buffer_Plan, Product_Root, Product, Forecast, Site_Plan, Active_Strategy, Location, Item, Skill, Item_Group, Request, Horizon, Site_Group, Product_Group, Delivery_Request, Strategy, Lot, Calendar_Entry, Calendar, Option, DAYS_OF_WEEK, DAY_OF_MONTH, DAY_OF_WEEK_OF_MONTH, DAY_OF_LAST_WEEK_OF_MONTH, User, Data_Directory, Report_Directory, Format, Date_Range, Quantity_Range, Horizon_Date, List, Model_Type, Field, Extension_Selector, Field_Error.

specification -- a Symbol field of model String

Specifies how to format Strings. A "\\$" in the format string specifies the place at which the string should be inserted.

Default: "\\$"

Symbol -- a spec extension of model Format

A Format which can handle Strings.

These models have a field that is a Symbol model :

Supply_Chain, Site, Seller, Plan, Problem_Category, Operation, Operation_Plan, Resource, Resource_Plan, Buffer, Product_Root, Product, Forecast, Location, Item, Skill, Item_Group, Request, Horizon, Site_Group, Product_Group, Delivery_Request, Strategy, Strategy_Lock, Item_Request, Load, Flow, Lot, Load_Size, SETUP, SKILL, Calendar_Entry, Calendar, SYMBOL, ACCESSORY, OPTION,

Extensions	Date Extension
------------	----------------

INCOMPATIBLE_ACCESSORY, INCOMPATIBLE_OPTION, EARLIEST, MPPS_Operation_Resource, REQUEST_FIXED, REQUEST_FIXED_WITH_ANALYSIS, Time_Table_Entry, Consolidation, CONSOLIDATION_OVERSIZE, CONSOLIDATION_UNDERSIZE, BLOCK_CYCLE, MPPS_Batch_Size, Dimension, Feature, Option, Option_Spec, CALENDAR_SYMBOL, SYMBOL, User, Format, Void, Logical, String, Symbol, Date, Date_Range, Number, Percentage, Integer, Quantity, Quantity_Range, Time, Restriction, List, Model_Type, Field, Extension_Selector.

specification - - a Symbol field of model Symbol
Specifies how to format Symbols. A "\\$" in the format string specifies the place at which the string should be inserted.
Default: "\$"

Date - - a spec extension of model Format

A Format which can handle Dates input or output in a variety of ways.

These models have a field that is a Date model :

Plan, Operation, Operation_Plan, Resource_Plan, Buffer_Plan, Problem, Active_Strategy, Operation_State, Request, Promise, Acceptance, Horizon_Bucket_Start, Delivery_Request, Delivery_Promise, Delivery_Acceptance, Forecast_Entry, Item_Request, Item_Acceptance, Delivery_Available_To_Promise, Item_Promise, Profile_Number, Profile_Percentage, Profile_Quantity, Yield_Changes, END, HOLD, Efficiency_Change, RAMP_LINEAR, REQUEST_ISSUED, PROMISE_DUE, REQUEST_DUE, DUE_DATE.

specification - - a Symbol field of model Date
Specifies how to format Date values. Special formatting characters are substituted with the corresponding date elements. All other characters will be part of the resulting date string.

The special formatting characters:

it == Hour in 12 hour format, leading zero

_l == Hour in 12 hour format, no leading zero

hh == Hour in 24 hour format, leading zero

Extensions	Date Extension
------------	----------------

_h == Hour in 24 hour format, no leading zero

mm == Minute, leading zero

_m == Minute, no leading zero

ss == Second, leading zero

_s == Second, no leading zero

AP == AM/PM flag, uppercase

ap == AM/PM flag, lowercase

zzz == date zone abbreviation

DD == Day of month, leading zero

_D == Day of month, no leading zero

MM == Month of year, leading zero

_M == Month of year, no leading zero

MMM == Month Abbreviation, all caps

Mmm == Month Abbreviation, capitalized

mmm == Month Abbreviation, lowercase

MR == Rounded Month of year, leading zero (month rounded up when day > 21)

_R == Rounded Month of year, no leading zero (month rounded up when day > 21)

MMR == Rounded Month Abbreviation, all caps

Mmr == Rounded Month Abbreviation, capitalized

mmr == Rounded Month Abbreviation, lowercase

YY == Year mod 100

Extensions	Date_Range Extension
------------	----------------------

YYYY == Year

WW == 2 character Day of week abbreviation, all caps

Ww == 2 character Day of week abbreviation, Capitalized

ww == 2 character Day of week abbreviation, lowercase

WWW == 3 character Day of week abbreviation, all caps

Www == 3 character Day of week abbreviation, Capitalized

www == 3 character Day of week abbreviation, lowercase

Default: MM/DD/YY hh:mm:ss

Date_Range -- a spec extension of model Format

A Format which can handle Date_Ranges

These models have a field that is a Date_Range model :

Plan, Operation_Plan, Product_Root, Problem, Delivery_Request, Delivery_Promise, Delivery_Acceptance, Forecast_Entry, ATP_Entry, Load_Plan, Flow_Plan, Item_Available_To_Promise, Product_Available_To_Promise, Calendar_Entry, Sorted_Bucket, Calendar_Plan.

specification -- a Symbol field of model Date_Range

Specifies how to format Date_Range values

Default: MM/DD/YY hh:mm:ss

separator -- a String field of model Date_Range

Specifies the separator between the min & max dates.

Default: /

Number -- a spec extension of model Format

A Format which can handle Integers and Numbers

These models have a field that is a Number model :

Extensions	Number Extension
------------	------------------

Site, Seller, Operation_Plan, Product, Problem, Active_Strategy, WEEKS, DAYS_WEEKS, DAYS, Delivery_Request, Delivery_Promise, Strategy, OPERATION_PLAN_RANK_RANGE, FEASIBILITY, MINIMIZE_PROBLEM_COUNT, MINIMIZE_PROBLEMS, MINIMIZE_LATENCY, WEIGHTED_LATENCY, MINIMIZE_SHORTNESS, WEIGHTED_SHORTNESS, MINIMIZE_COST, MAXIMIZE_PROFIT, MAXIMIZE_REVENUE, Active_Problem, Active_Goal, FEASIBILITY, MINIMIZE_PROBLEM_COUNT, MINIMIZE_PROBLEMS, MINIMIZE_LATENCY, WEIGHTED_LATENCY, WEIGHTED_SHORTNESS, MINIMIZE_COST, MAXIMIZE_PROFIT, MAXIMIZE_REVENUE, Alternate_Product, Profile_Number, Ordered_Sub_Strategy, SEQUENCE_RUN_ONCE, SEQUENCE_RUN_MULTIPLE, Box, Calendar_Entry, NUMBER, NUMBER_QUANTITY, MAX, MIN_MAX, TIME_MULTIPLE, MPPS, Operation_Resource, Routing_Operation, MAX_SETUPS_PERIOD, Flow_Criterion, CUSTOMER_RANK, SELLER_RANK, REQUEST_RANK, PROMISE_RANK, ACTUAL_OR_FORECAST, REQUEST_ISSUED, PROMISE_DUE, REQUEST_DUE, DUE_DATE, PROMISED_ENTRY_DATE, NUMBER, NUMBER_QUANTITY, NUMBER, NUMBER_QUANTITY, Data_Directory, Report_Directory.

specification -- a Symbol field of model Number

1) A number format is specified using '0', '#' and '?'. A '0' puts that digit or '0' if there is no number at that position. A '?' puts the number at that position or a '(space)' if there is no number at that position. A '#' puts the number at that position if there is a number, else it does not put anything. A decimal point (.) separates the decimal part from the fractional part. The total number of zero + pound + question marks to the right of the point denotes the maximum number of digits to be displayed after the point. There can be characters within the decimal part of the number. 2) Anything within double quotes are copied directly to the output. 3) Single characters '\$', ',', '.', '(', ')', '-' in the format are copied directly to the output. 4) Any character after a 'h' is displayed only if the number is negative. Otherwise a space is displayed there. If there is no 'h' in the format, negative numbers will be displayed without any '-' sign. See Example (4). 5) Any character after a backslash (\) is displayed as it is. 6) A comma at any place in the format means that the number should be displayed in comma notation. 7) A number of blanks may be specified using \$xx, where xx is the 2 digit number of blanks. Note that xx *must* be 2 digits. This feature may be used to provide room in the string for the units to be added later. Example 2 specifies an 8 character buffer to follow the number.

```

Examples: ----- format_ = n%(##) exp_format_(compiled format_) =
(#####)##### Input number: 2,342,94e+08 is displayed as 2,342,943,12,23
Input number: -1,234,23e+06 is displayed as (1,234,234,1) Input number: 1,234,23e+06
is displayed as 1,234,234,12 Input number: 1,234,2e+07 is displayed as
2,344,234,13

```

Percentage -- a spec extension of model Format

A Formal which can handle Percentages

These models have a field that is a Percentage model

Operation_Plan, Resource_Plan, Delivery_Promise, Strategy, Problem_Set, Strategy_Change, Strategy_Goal, Active_Problem, Sub_Product_Group, Sub_Product, Item_Promise, Profile_Percentage, Ranked_Sub_Strategy, MAINTENANCE, Resource_Skill, Skill_Resource, PER_ALLOCATED, PER_COMMITTED, MEMBER_RANK, FIXED_SPLIT, Product_Allocation, Alternate_Operation, Effective_Calendar_Operation, PRODUCE_YIELD, PRODUCE_YIELD_CALENDAR, Yield_Changes, MIN_MAX, MPPS, MPPS_Operation_Resource, REWORK, Consolidation, CONSOLIDATION_OVERSIZE, CONSOLIDATION_OVERSIZE, CONSOLIDATION_UNERSIZE, CONSOLIDATION_UNERSIZE, FIXED_Efficiency_Change, RAMP_LINEAR, SHARED_USE, OVERSIZE, OVERSIZE, BUCKET_OVERSIZE, BUCKET_OVERSIZE, UNDERLOAD, UNDERLOAD, CUMULATIVE_OVERLOAD, CUMULATIVE_OVERLOAD, FIXED, FIXED, BUCKETED_NESTED_SORT, BUCKETED_COMBINED_SORT, Option_Spec, CALENDAR.

Integer -- a spec extension of model Format

A Format which can handle Integers

These models have a field that is a Integer model :

Site, Seller, Operation, Operation_Plan, Resource, Resource_Plan, Buffer, Buffer_Plan, Product_Root, Product, Forecast, Site_Plan, Problem, Active_Strategy, LINK, Item, Skill, Configuration, SUPPLIER, CUSTOMER, LINK, Operation_State, Request, Promise, SUPPLIER_DAYS, LIST_HORIZON_DATES, DATES, MONTHS_WITH_FULL_WEEKS, Site_Group, Delivery_Request, Delivery_Promise, Delivery_Acceptance, Strategy, Problem_Set, Strategy_Change, Strategy_Look, Strategy_Goal, OPERATION_PLAN_RANK_RANGE, OPERATION_PLAN_RANK_EXPRESSION, Active_Goal, Forecast_Entry, Load, Flow, Operation_Problem_Detector, Buffer_Problem_Detector, USE_ALTERNATE_OPERATION,

Extensions

Integer Extension

REQUEST_NOT_PLANNED, REQUEST_PLANNED_LATE,
REQUEST_PLANNED_EARLY, REQUEST_PLANNED_SHORT,
REQUEST_PLANNED_EXCESS, PROMISE_NOT_PLANNED,
PROMISE_PLANNED_LATE, PROMISE_PLANNED_EARLY,
PROMISE_PLANNED_SHORT, PROMISE_PLANNED_EXCESS,
ACCEPTANCE_NOT_PLANNED, ACCEPTANCE_PLANNED_LATE,
ACCEPTANCE_PLANNED_EARLY, ACCEPTANCE_PLANNED_SHORT,
ACCEPTANCE_PLANNED_EXCESS, SEQUENCE_RUN_ONCE,
SEQUENCE_RUN_MULTIPLE, SEQUENTIAL_ALTERNATES,
SEQUENTIAL_ALTERNATES,
SEQUENTIAL_ALTERNATES_KEEP_BEST,
PROPORTIONAL_ALTERNATES, KEEP_BEST,
PROPORTIONAL_RESOLVES, ORDERED_RESOLVES, Load, Size,
PRECEDENCE, OVER_RESTRICTION, EXPEDITED,
PLANNED_BEFORE_CURRENT, UNRELEASED, NEEDS_RELEASE,
INCONSISTENT_OPLAN_OPERATION, REQUEST_PROMISED_LATE,
REQUEST_PROMISED_EARLY, REQUEST_PROMISED_SHORT,
REQUEST_PROMISED_EXCESS, PROMISE_NOT_OFFERED,
PROMISE_NOT_CONFIRMED, PROMISE_NOT_ACCEPTED,
ACCEPTANCE_INCONSISTENT, REQUEST_QUEUED, REQUEST,
REQUEST_PLAN, PROMISE, PROMISE_PLAN, REQUEST_PROMISE,
DELIVERY_REQUEST_NOT_COORDINATED,
DELIVERY_PROMISE_NOT_COORDINATED,
DELIVERY_ACCEPTANCE_NOT_COORDINATED, Resource_Skill,
Resource_Problem_Detector, Skill_Resource,
PROPORTIONAL_INTERACTION, EARLIEST_PROBLEM_START,
SORT_BY_EXPRESSION, SUPPLY_PLANNED_LATE,
SUPPLY_PLANNED_EARLY, SUPPLY_PLANNED_SHORT,
SUPPLY_PLANNED_EXCESS, SUPPLY_PROMISED_LATE,
SUPPLY_PROMISED_EARLY, SUPPLY_PROMISED_SHORT,
SUPPLY_PROMISED_EXCESS, SUPPLY, SUPPLY_PLAN,
SUPPLY_PROMISE, FCFS, PER_ALLOCATED, PER_COMMITTED,
MEMBER_RANK, FIXED_SPLIT, CALENDAR_SPLIT, SLIDING,
HORIZON, BUCKETED_ALL, BUCKETED_ASAP, FIXED,
AT_END, SIMPLE_FIXED_QUANTITY, SINGLE_REQUEST,
SIMPLE_FIXED_TIME, WEEKLY, DUAL_REQUEST, ON_TIME,
ON_TIME, ON_TIME, FULL_QUANTITIES_OF_ALL_ITEMS,
FULL_QUANTITIES_OF_ALL_ITEMS,
UNRESTRICTED, UNRESTRICTED, UNRE-
STRICTED, UNRESTRICTED, NUMBERED, NONE, NONE, FIXED,
ON_TIME, ALL, ALL, ON_TIME, ASAP, ASAP_MONTHLY,
BUCKETED_ALLOCATION, BUCKETED_ALL_MIN_PRICE,

Extensions

Integer Extension

BUCKETED_MIN_PRICE_ASAP, SHIP_IN_RATIO, Calendar_Entry,
Calendar, UNSPECIFIED, NUMBER, QUANTITY,
NUMBER_QUANTITY, SYMBOL, TIME, ALTERNATES_PRIMARY,
Alternate_Operation, ALTERNATES_PROPORTIONAL,
EFFECTIVE_CALENDAR, CONSUME_FIXED, CONSUME_PER,
PRODUCE_FIXED, PRODUCE_PER, PRODUCE_YIELD,
PRODUCE_YIELD_CALENDAR, PRODUCE_YIELD_RAMP_LIST,
PRODUCE_YIELD_RAMP_CALENDAR, EARLIEST_FIXED, LINEAR,
RESOURCE, ONE, MAX, MIN, MAX, UNIDENTIFIED_OP_STATE,
DELAY_ONLY_FIXED, DELAY_ONLY_BASIC,
DELAY_ONLY_CALENDAR, BASIC_CALENDARS, FIXED_TIME,
TIME_MULTIPLE, BASIC, BASIC_DELAYED, FILL, OFFSET,
TIME_EXPRESSION, REQUEST_FIXED,
REQUEST_FIXED_WITH_ANALYSIS, ROUTING, STARTED, COM-
PLETED, IN_FRONT, SIMPLE_CONSOLIDATION, UNCONSOL-
DATED, UNCOORDINATED, CONSOLIDATION_OVERSIZE,
CONSOLIDATION_UNDESIZE, FIXED, STEP_LIST, RAMP_LINEAR,
RAMP_LIST, CALENDAR, RAMP_CALENDAR, LOAD_TIME,
WEIGHTED_TIME, INFINITE_USE, EXCLUSIVE_USE, SHARED_USE,
ZERO, OVERLOAD, OVERTIME, OVERSIZE, BUCKET_OVERSIZE,
UNDERLOAD, CUMULATIVE_OVERLOAD, RESOURCE, FIXED,
CALENDAR_PRIMARY, PREFER_PRIMARY, EVEN,
MAX_EFFICIENCY, UNLIMITED, FIXED_COUNT,
FIXED_QUANTITY, CALENDAR_COUNT, CALENDAR_QUANTITY,
MULTI_DIMENSION, CALENDAR_RATE, FIXED, CALENDAR,
ZERO, FIXED, NEGATIVE_ON_HAND, OVER_FLOW_LIMIT,
NEGATIVE_ON_HAND_AT_END, LOT_OVER_CONSUMED,
LOT_NOT_CONSUMED, LOT_NOT_PRODUCED,
EXCESS_ON_HAND_PRODUCED, LOW_ON_HAND, EXCESS_ON_HAND,
Flow_Criterion, BUCKETED_COMBINED_SORT,
PRODUCING_FLOW_CALENDAR,
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK,
SUPPLY_CALENDAR, ON_HAND_CALENDAR,
ON_HAND_CALENDAR_FILTER_AND_RANK,
FLOW_LIMIT_CALENDAR,
FLOW_LIMIT_CALENDAR_FILTER_AND_RANK, CUSTOMER_RANK,
SELLER_RANK, REQUEST_RANK, ACTUAL_OR_FORECAST,
REQUEST_ISSUED, PROMISE_DUE, REQUEST_DUE, DUE_DATE,
PROMISED, ENTRY_DATE, INFINITE, SUPPLIER, FLOW_THRU,
PHANTOM, BASIC, BASIC_FILTER_AND_RANK, FIXED_QUANTITY,
MULTIPLE, MULTIPLE_FILTER_AND_RANK,

Extensions	Quantity Extension
------------	--------------------

FIXED_QUANTITY_FENCED, FIXED_TIME, STANDARD, CUSTOM, LFL_SIMPLE, LFL_BOUNDED, MFL_BOUNDED, MANUAL, CALENDAR, Calendar_Plan, EVERYDAY, EVERY_N_DAYS, WEEKDAYS, WEEKENDS, DAYS_OF_WEEK, DAY_OF_MONTH, DAY_OF_WEEK_OF_MONTH, DAY_OF_LAST_WEEK_OF_MONTH, DAY_OF_YEAR, YEARLY, Format, Void, Logical, String, Symbol, Date, Date_Range, Number, Percentage, Integer, Quantity, Quantity_Range, Time, Restriction, Horizon_Date, List, Field.

specification -- a Symbol field of model Integer
Specifies how to format Integers. See NUMBER specification.
Default: %d

Quantity -- a spec extension of model Format

A Format which can handle Quantities.

These models have a field that is a Quantity model:

Operation_Plan, Resource_Plan, Buffer_Plan, Product_Root, Product_Location, LINK, Strategy, Unit, Unit_Quantity, Generic_Product, Alternate_Product, Sub_Product_Group, Sub_Product_Forecast_Entry, ATP_Entry, Item_Request, Flow, Load_Plan, Flow_Plan, Item_Acceptance, Item_Promise, Item_Available_To_Promise, Product_Available_To_Promise, Lot_Flow, Profile_Number, Profile_Percentage, Profile_Quantity, Lot_Request_PLANNED_SHORT, REQUEST_PLANNED_EXCESS, PROMISE_PLANNED_SHORT, PROMISE_PLANNED_EXCESS, ACCEPTANCE_PLANNED_SHORT, ACCEPTANCE_PLANNED_EXCESS, REQUEST_PROMISED_SHORT, REQUEST_PROMISED_EXCESS, SUPPLY_PLANNED_SHORT, SUPPLY_PLANNED_EXCESS, SUPPLY_PROMISED_SHORT, SUPPLY_PROMISED_EXCESS, MEMBER_RANK, SIMPLE_FIXED_QUANTITY, SINGLE_REQUEST, SIMPLE_FIXED_TIME, WEEKLY, DUAL_REQUEST, Calendar_Entry, QUANTITY, NUMBER_QUANTITY, Alternate_Operation, Effective_Calendar_Operation, CONSUME_FIXED, CONSUME_PER, PRODUCE_FIXED, PRODUCE_PER, PRODUCE_YIELD, PRODUCE_YIELD_CALENDAR, FIXED_LINEAR, Routing_Operation, REMAINING_TIME, COMPLETED_TIME, PERCENT_COMPLETE, CUMULATIVE, ARRIVED, STARTED, SCRAPPED, COMPLETED, ON_HAND, IN_FRONT, IN_PROCESS, IN_BACK, Unordered_Operation,

Extensions	Quantity_Range Extension
------------	--------------------------

MPS_BATCHING, FIXED_QUANTITY, Size_Dimension, MIN_ON_HAND, MAX_ON_HAND, NEGATIVE_ON_HAND, NEGATIVE_ON_HAND_OVER_FLOW_LIMIT, OVER_FLOW_LIMIT, NEGATIVE_ON_HAND_AT_END, NEGATIVE_ON_HAND_AT_END, LOT_OVER_CONSUMED, LOT_OVER_PRODUCED, LOW_ON_HAND, LOW_ON_HAND_EXCESS_ON_HAND, EXCESS_ON_HAND, EXCESS_ON_HAND_AT_END, EXCESS_ON_HAND_AT_END, BUCKETED_NESTED_SORT, Sorted_Bucket, BUCKETED_COMBINED_SORT, CALENDAR_PRODUCING_FLOW_CALENDAR, PRODUCING_FLOW_CALENDAR, PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK, PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK, ON_HAND_CALENDAR, ON_HAND_CALENDAR_FILTER_AND_RANK, FLOW_LIMIT_CALENDAR, FLOW_LIMIT_CALENDAR_FILTER_AND_RANK, FLOW_THRU, FLOW_MIN_TIME, FLOW_MIN_ON_HAND, BASIC_BASIC, BASIC_FILTER_AND_RANK, BASIC_FILTER_AND_RANK, FIXED_QUANTITY, FIXED_QUANTITY, MULTIPLE, MULTIPLE, MULTIPLE_FILTER_AND_RANK, MULTIPLE_FILTER_AND_RANK, FIXED_QUANTITY_FENCED, FIXED_QUANTITY_FENCED, FIXED_TIME, FIXED_TIME, CALENDAR, QUANTITY, NUMBER_QUANTITY, QUANTITY, NUMBER_QUANTITY.

specification -- a Symbol field of model Quantity
Specifies how to format Quantities. This is a combination of a NUMBER specification and a Measure_Unit specification.
Default: n-##-## S3

Quantity_Range -- a spec extension of model Format

A Format which can handle Quantity Ranges.

These models have a field that is a Quantity_Range model:
Item_Request, Item_Acceptance, Item_Promise, PRODUCING_FLOW_CALENDAR, PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK, MULTIPLE, MULTIPLE_FILTER_AND_RANK, LFL_BOUNDED, MFL_BOUNDED, specification -- a Symbol field of model Quantity_Range
Specifies how to format Quantity_Range values
Default: n-##-## S3

separator - - a String field of model Quantity. Range
Specifies the separator between the min & max quantities.
Default: <

Time -- a spec extension of model Format

A Format which can handle Times.

These models have a field that is a Time model:

Operation_Plan, Resource_Plan, Product_Root, Product_Active_Strategy,
Delivery_Request, Delivery_Promise, Delivery_Acceptance, Strategy,
Problem_Set, Generic_Product, Alternate_Product, Item_Request,
Item_Acceptance, Item_Promise, REQUEST_PLANNED_LATE,
REQUEST_PLANNED_EARLY, PROMISE_PLANNED_LATE,
PROMISE_PLANNED_EARLY, ACCEPTANCE_PLANNED_LATE,
ACCEPTANCE_PLANNED_EARLY, PRECEDENCE,
REQUEST_PROMISED_LATE, REQUEST_PROMISED_EARLY,
SUPPLY_PLANNED_LATE, SUPPLY_PLANNED_EARLY,
SUPPLY_PROMISED_LATE, SUPPLY_PROMISED_EARLY,
SIMPLE_FIXED_TIME, Calendar_Entry, TIME, PRODUCE_YIELD,
PRODUCE_YIELD_CALENDAR, PRODUCE_YIELD_RAMP_LIST,
PRODUCE_YIELD_RAMP_CALENDAR, DELAY_ONLY_FIXED,
DELAY_ONLY_BASIC, FIXED_TIME, TIME_MULTIPLE, BASIC,
BASIC_DELAYED, MPPS, REQUEST_FIXED,
REQUEST_FIXED_WITH_ANALYSIS, Request_Alternates,
SETUP_FIXED, SETUP_TABLE, Time_Table_Entry,
SETUP_CALENDAR, SETUP_EXPRESSION, SKILL_FIXED,
SKILL_TABLE, SKILL_CALENDAR, SKILL_EXPRESSION,
TRANSIT_FIXED, TRANSIT_TABLE, TRANSIT_CALENDAR,
TRANSIT_EXPRESSION, SHARED_USE, MPPS_BASIC, LOAD_TIME,
DUAL_WEIGHTED_TIME, MAX_SETUPS_PERIOD,
MAX_SETUP_TIME_PERIOD, OVERLOAD, OVERLOAD,
TIME, OVERTIME, FIXED, MTBF, LOT_OVER_CONSUMED,
LOT_OVER_PRODUCED, LOT_OVER_PRODUCED,
PRODUCING_FLOW_CALENDAR,
PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK,
FLOW_MIN_TIME, BASIC, BASIC_FILTER_AND_RANK,
FIXED_QUANTITY, MULTIPLE, MULTIPLE_FILTER_AND_RANK,
FIXED_QUANTITY_FENCED, FIXED_TIME, LFL_MIN_TIME, CAL-
ENDAR, TIME, TIME.

specification . . . a Symbol field of model Time
Specifies how to format Time values. This can be as a Quantity (see QUANTITY) or
in hour-minute-second form. The letter 'h' is substituted for hours, the letter 'm' for
minutes, and the letter 's' for seconds (similar to the DATE format).
Default: hh:mm

Restriction -- a spec extension of model Format

A Format which can handle Restriction objects (see the documentation for the restric-
tion type). The format is:

```
input      period start end within latest  output
=====
unspecified (00.00) f f f f f [unspecified]
start before <date> (00.d) f f f f f start before <date>
start after <date> (d.00) f f f f f start after <date>
after <date>
Restriction(d,ime) (00.d) f f f f f start after <date>
end before <date> (00.d) f f f f f end before <date>
before <date>
Restriction(d,false) (00.d) f f f f f end before <date>
end after <date> (d.00) f f f f f end after <date>
end after <date>
(d1,d2) f f f f f end first in <period>
end first in <period>
(d1,d2) f f f f f end last in <period>
end first in <period>
(d1,d2) f f f f f end first in <period>
end last in <period>
end last in <period>
last in <period>
(d1,d2) f f f f f start first in <period>
```

Extensions	Horizon, Date Extension
------------	-------------------------

```

.....
(d1,d2) t f t start last in <period>
.....
start first in <period> (d1,d2) t f t start first in <period>
start in <period>
first in <period>
.....
start last in <period> (d1,d2) t f t start last in <period>
.....
not in <period> (d1,d2) t t f f not in <period>
.....
(d1,d2) t t f t not in <period>
.....
in <period> (d1,d2) t t t f in <period>
.....
(d1,d2) t t t t

```

These models have a field that is a Restriction model:
Operation_Plan, Load_Plan.

specification - - a *Symbol field of model Restriction*
Specifies how to format Restriction values. The specification is identical to Date formats.
Default: YY-MM-DD hh:mm

Horizon_Date - - a *spec extension of model Format*

Specifies how to format a Horizon_Date value.

These models have a field that is a Horizon_Date model:
Operation, Strategy, Problem_Set, SIMPLE_FIXED_QUANTITY,
SIMPLE_FIXED_TIME, WEEKLY, DUAL_REQUEST,
SIMPLE_CONSOLIDATION, SHARED_USE,
PRODUCING_FLOW_CALENDAR,
FIXED_QUANTITY_FENCED, LFL_BOUNDED, MLFL_BOUNDED,
LFL_MIN_TIME.

time_format - - a *String field of model Horizon_Date*
Specifies the format of the time portion of the horizon_date. This is the same format as described for the time portion of Date, except the granularity for horizon_date times goes only to minutes (seconds are ignored).

Extensions	Horizon, Date Extension
------------	-------------------------

Default: hh:mm

number_format - - a *String field of model Horizon_Date*
The number format determines whether numbers are printed as numeric or ordinal values (i.e. whether 'first' should be printed as '1', '1st' or 'first'). The following character sets are supported to define the format:

```

'%' (digits - i.e. '1') -or- '%st' (ordinal - i.e. '1st') -or- '%t' (ordinal - full name - i.e. 'first')
Default: #st

```

weekday_format - - a *String field of model Horizon_Date*
The day_of_week format is used to print the day of the week. The supported formats are as follows: # (digit - i.e. day '1') -or- '#st' (ordinal - i.e. '1st day') -or- 'st' (full name - i.e. 'first day') -or- 'WW' - 2 character Day of week abbreviation, all caps (i.e. 'MO') 'Ww' - 2 character Day of week abbreviation, Capitalized 'ww' - 2 character Day of week abbreviation, lowercase 'WWW' - 3 character Day of week abbreviation, all caps 'Www' - 3 character Day of week abbreviation, Capitalized 'www' - 3 character Day of week abbreviation, lowercase 'WWW*' - full Day of week name, all caps (i.e. 'MONDAY') 'Www*' - full Day of week name, Capitalized 'www*' - full Day of week name, lowercase
Default: Www

day_format - - a *String field of model Horizon_Date*
Nth day of horizon format. Keywords beginning with ':' are replaced by the actual value. :DAY is required :TIME is optional
Default: :DAY day

rel_day_of_month_format - - a *String field of model Horizon_Date*
[1-31]th day of Nth month of horizon format. Keywords beginning with ':' are replaced by the actual value. :DAY, :MONTH are required :TIME is optional
Default: :DAY of :MONTH month

day_of_month_format - - a *String field of model Horizon_Date*
[1-31]th day from today of Nth month of horizon format. Keywords beginning with ':' are replaced by the actual value. :DAY, :MONTH are required :TIME is optional
Default: :DAY day and :MONTH month

day_from_end_month_format - - a *String field of model Horizon_Date*
[1-31]th day from last day of Nth month of horizon format. Keywords beginning with ':' are replaced by the actual value. :DAY, :MONTH are required :TIME is optional
Default: :DAY from end of :MONTH month

Extensions	List Extension
------------	----------------

DOW_format - - *a String field of model Horizon_Date*
Nth occurrence of [1-7]th day of horizon format. Keywords beginning with ':' are replaced by the actual value. :OCCURRENCE, :DOW are required :TIME is optional
Default: :OCCURRENCE :DOW

DOW_of_week_format - - *a String field of model Horizon_Date*
[1-7]th day of Nth week of horizon where week starts on day [1-7] format. Keywords beginning with ':' are replaced by the actual value. :DOW, :WEEK are required :WEEK_START, :TIME are optional
Default: :DOW of :WEEK week, start :WEEK_START

DOW_of_month_format - - *a String field of model Horizon_Date*
Nth occurrence of [1-7]th day of Nth month of horizon format. Keywords beginning with ':' are replaced by the actual value. :OCCURRENCE, :DOW, :MONTH are required :TIME is optional
Default: :OCCURRENCE :DOW of :MONTH month

DOW_of_week_month_format - - *a String field of model Horizon_Date*
[1-7]th day of [1-4]th week of Nth month of horizon where week starts on day [1-7] format. Keywords beginning with ':' are replaced by the actual value. :DOW, :WEEK are required :WEEK_START, :TIME are optional
Default: :DOW, :WEEK wk, :MONTH month, start :WEEK_START

List - - a spec extension of model Format

A Format which can handle all List types

delimiter - - *a String field of model List*
Specifies the separator string to place between list elements. It is not put after the last element.
Default: .

element_format - - *a Symbol field of model List*
The format to use when formatting the list elements
Default: default

width - - *a Integer field of model List*
The result when converting a list into a string will be less than this width, in characters. If the result is truncated, a '.' will be at the end of the string. For example: width = 20; list(1, 2, 3) -> "1, 2, 3" width = 10; list(1, 2, 3) -> "1, 2..."
Default: 100

Extensions	List Extension
------------	----------------

truncated_format - - *a String field of model List*
If the result is truncated due to the width field, this specifies the format of the stuff at the end of the result. '{0}' will contain the count of the list elements which have not been formatted. '{1}' will contain the data type of the list element. For example: if max_end_format = "...{(0)}", and width = 10, then list(1, 2, 3) -> "1, 2,...{1}"

if max_end_format = "[list of {0} {1}]s'", and width = 1, then list(1, 2, 3) -> "[List of 3 Integer]s"
Default: (+ {0}...)

10.46 Field Extensions

10.46.1 field_type extensions of model Field

SIMPLE -- *a field_type extension of model Field*

A builtin field of a builtin model

SELECTOR -- *a field_type extension of model Field*

A builtin field that selects a set of extension fields.

EXTENDED -- *a field_type extension of model Field*

A builtin field that is only present in a subset of it's parent's models

USER -- *a field_type extension of model Field*

A user defined field

get_expression -- *a Expression field of model USER*

When the 'get_expression' field is non-empty, this user-defined field will compute it's value from the expression. When the expression is evaluated, '#' will be bound to an instance of this field's model. The expression will be expected to return a the data-type indicated by this field's 'type' field.

@-Note: A Field with a get_expression, but no set_expression is read-only.

Note: The default field has no effect for fields with a get_expression.

Note: A user-defined field normally consumes several words of storage for each model instance. That's not true for a computed field.

Default: none

set_expression -- *a Expression field of model USER*

When the 'set_expression' field is non-empty, this user-defined field will execute the given expression when the field is set. 'self' will be bound to an instance of this field's model, and 'value' will be set to the new value (the 2nd parameter to the set function).

The result from the 'set_expression' is ignored. A field with a 'set_expression' but no 'get_expression' will generate a Field_Error.

Default: none

Index

Symbol . Date Range)	354, 718	active_product_000_forecast	450
A		active_products	310
abs Integer	78	active_specific_forecast	456
abs Number	78	active_specific_products	332
abs Quantity	79	active_strategies	348
abs Time	79	Active Strategy	230, 348, 483, 484, 485, 486, 487, 488, 489, 491, 560, 786, 787, 788, 789, 790, 791
accept, as allocated	432, 437, 472	List	348, 787, 790
accept, as promised	439, 441, 445	active_sub_forecast	458, 460, 714
accept, by	391, 413, 438	active_top_forecast	450
Acceptance	229, 352, 355, 413, 437, 438, 439, 440, 443, 717, 719, 721, 723, 724, 726, 728, 729, 731, 732, 734, 736, 737, 739, 741, 743, 744, 747, 748, 749, 751, 752, 753, 754, 755, 756, 757, 758, 759, 763, 765, 767, 768, 770, 771, 772, 773, 775	active_ui	534
acceptance	413, 721, 723, 724, 726, 728, 729, 731, 732, 734, 736, 737, 739, 741, 743, 744, 747, 748, 749, 751, 752, 753, 754, 755, 756, 757, 758, 759, 763, 765, 767, 768, 770, 772, 773, 775	actual	398
ACCEPTANCE_INCONSISTENT	559, 564, 755, 834, 835	ACTUAL_OR_FORECAST	507, 871
ACCEPTANCE_NOT_PLANNED	559, 563, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 775	actual_promises	452, 473
ACCEPTANCE_PLANNED_EARLY	559, 563, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 763, 765, 767, 768, 770, 772, 773, 775	actual_requests	451
ACCEPTANCE_PLANNED_EXCESS	559, 563, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 775	adjusted_target	491
ACCEPTANCE_PLANNED_LATE	559, 563, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 775	adjusted_value	490
ACCEPTANCE_PLANNED_SHORT	559, 563, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 775	after_fence_excess_on_hand	625, 630
acceptances	352, 355, 717, 719	after_fence_multiple_quantity	664
accepted	395, 438, 471	after_fence_quantity	625, 629
access	546, 548	after_fence_number	875, 876
active	326, 455	after_horizon	877
active_forecast	450	after_horizon_quantity	876, 877
active_generic_forecast	456	after_horizon_symbol	878
Active Goal	230, 487, 490, 491, 561, 792, 793, 794	after_horizon_time	878, 815
active_goals	487	after_resolve	788, 815
active_leaf_product_forecast	458, 460, 714	after_run	788, 815
active_member_forecast	456	after_search	788, 815
active_primary_products	332	after_set	548
Active Problem	230, 486, 488, 489	after_success	788, 813
active_problems	486	ALL	562, 804
active_product_forecast	451	all_consume_flow	283
		all_consuming_flow_plans	379
		all_consummg_operations	238
		all_flow	283
		all_items	304
		ALL_OR_TIME	562, 804
		all produce flows	283
		all producing flow plans	379
		all producing operations	258
		all_products	339
		all_products_and_specifies	340
		all_sic_group	309
		all_supplying_operations	259
		allocac_allocated_available	457, 476

allocac, excess	677	BASIC_CALENDARS	689
allocac, queue, multiple	330	BASIC_DELAYED	556, 588, 589, 607
allocac, available	470, 478	BASIC_FILTER_AND_RANK	558, 557, 590, 591, 607
allocac, policy	327, 476, 580, 581, 583, 715	BEFORE_AND_AFTER	561, 563, 787, 788, 814, 815
allocations	583	before, horizon, number	875, 876
Allocac, Operation	228, 297, 298, 586, 587	before, horizon, quantity	875, 877
Allocac, Product	229, 332, 335	before, horizon, symbol	877
allocac, products	332	before, horizon, time	878
allocac, units	586, 587	before, run	787, 815
ALTERNATES_PRIMARY	555, 556, 586, 605, 606	before, search	787, 815
ALTERNATES_PROPORTIONAL	555, 556, 586, 587, 606	billing, address	240
always, override, members, committed	328	billing, city	240
always, override, members, forecasted	328	billing, contact	240
and, Logical	143	billing, country	240
announcing, goodness	485	billing, fax	240
area	517	billing, name	239
artificial	251	billing, phone	240
ASAP	562, 804	billing, postal_code	240
ASAP_MONTHLY	562, 805	billing, state	240
AT_END	555, 580	box	247
at, entries	459, 713	break, expression, symbol, expression	163
ATP_Entry	230, 459, 478, 479, 713	breakpoint	231, 534
atp, horizon	311, 451	breakpoint, symbol	164
auto, allocac	327	bucket, name	615
auto, allocac, from, organization	331	bucket, list, by, date, list, void, date	60
auto, run	485	bucket, list, by, key, list, void, symbol	59
auto, run, strategy	348	BUCKETED_ALL	562, 804
availability, horizon	584	BUCKETED_ALL_MIN_PRICE	562, 805
availability, policy	328, 584	BUCKETED_ALLOCATION	562, 805
availability, policy	434, 474	BUCKETED_ASAP	562, 806
available, dates	478	BUCKETED_MIN_PRICE_ASAP	562, 806
available, sized, time (Date, Range)	369	BUCKETED_NESTED_SORT	557, 558, 621, 622, 623, 677
available, time (Date, Range)	366	bucket, override	779
available, to, premise	473	bucket, spec	493, 810
average, list, integer	106	bucket, symbol, list, void	811
average, list, number	107	BUCKETED_ALL	562, 804
average, list, percentage	107	BUCKETED_ALL_MIN_PRICE	562, 805
average, list, quantity	108	BUCKETED_ALLOCATION	562, 805
average, list, time	108	BUCKETED_ASAP	562, 806

background, expression	157	consolidation	776, 777
background, run, strategy	486	consolidation, name	614
balance, sized, time (Date, Range, Logical, Logical)	348	CONSOLIDATION_OVERSIZE	560, 564, 776, 777, 846
balance, sized, time (Date, Range, Logical, Logical)	370	CONSOLIDATION_UNDESIZE	560, 564, 777, 846
balance, sized, time (Date, Range, Logical, Logical)	370	consolidations	619
balance, sized, time (Date, Range, Logical, Logical)	370	CONSUME	536, 604, 609
balance, sized, time (Date, Range, Logical, Logical)	368	consume, earlier	329
base, time	590	CONSUME_FIXED	556, 598
base, units	590	consume, flows	283
BASIC	356, 357, 358, 590, 607, 648, 649, 686, 687, 688, 361, 362, 363, 364, 383, 558, 603, 677, 681, 685, 686, 689	CONSUME_PER	556, 598
		consumed	472, 478
		consumed, forecast	427
		consumed, flow (Date, Range)	337
		consuming, flow, plan	385
		consuming, flow, plans	357, 364
		consuming, flows	362
		consuming, operation	667, 669, 671
		cont, integer	164
		contains, list, void	49
		continue, row, plan, selection	635, 641, 645, 652, 661
		Control (Measure, Unit, Measure)	230, 530
		convert, quantity, quantity	512
		cost, number	80
		cost, list, void	261, 296, 480
		create, consuming, operation, plan, measure, unit, restriction	359
		create, producing, operation, plan, measure, unit, restriction	359
		criteria	359
		criticism	623
		cumulative, accepted	471
		cumulative, allocated	470
		cumulative, allocated, available	475, 479
		cumulative, available	474
		cumulative, committed	465
		cumulative, consumed	462
		cumulative, forecasted	462
		cumulative, planned	468
		cumulative, planned, available	475
		current, all	346
		current, data, directory	533
		current, depth, cell	66
		current, index, cell	145
		current, sub	787
		CUSTOMER	561, 562, 796, 799
		CUSTOMER	228, 229, 245, 355, 555, 559, 570, 719
		customer, plan	317, 325
		CUSTOMER_RANK	393, 399
		customer, service, level	566, 870
		customers	318, 325
		cycle, on, hand (Date)	679, 683, 688, 692, 696, 699, 703,

Index

[illegible]

Index

item, acceptance, filter	826, 827, 828
item, acceptances	441
item, and	425, 427
item, available, To, Promise	229, 427, 433, 434, 435, 437
List	425
Item_Group	228, 243, 302, 303, 304, 305, 306, 569
List	243, 569
item, group	305
item, group	243, 569
item, name	592, 594
item, Promise	229, 407, 418, 426, 427, 428, 429, 430, 431, 432, 433, 434, 444, 604, 720, 722, 724, 725, 727, 729, 730, 732, 733, 735, 737, 738, 740, 742, 743, 749, 750, 751, 762, 764, 766, 768, 769, 771, 772, 774
List	473, 476
item, promise	407, 444, 720, 722, 724, 725, 727, 729, 730, 732, 733, 735, 737, 738, 740, 742, 743, 749, 750, 751, 762, 764, 766, 768, 769, 771, 772, 774
item, promise, filter	823, 824, 825, 833, 837, 843
ITEM, PROMISE, OVERPRICED	559, 751, 752
item, promise	418
item, request	229, 398, 405, 406, 407, 408, 409, 410, 411, 426, 604, 608, 720, 722, 724, 725, 727, 728, 730, 732, 733, 735, 737, 738, 740, 742, 743, 749, 750, 751, 762, 764, 766, 768, 769, 771, 772, 774
List	354, 476, 718
item, request	426, 608, 720, 722, 724, 725, 727, 728, 730, 732, 733, 735, 737, 738, 740, 742, 743, 749, 750, 751, 762, 764, 766, 768, 769, 771, 772, 774
item, request, filter	820, 821, 822, 836, 839, 840, 841, 844
item, request	398
items	243, 244, 320, 324, 569, 570
justify, String, int, String	71
key, jname, List, Void	56
key, Symbol, Logical, Logical	54
key, value, List, Void, Symbol	56
keyed element, Symbol	63
keyed list, List, Void, Expression	62
keys, List, Void	63
language	535
last, change	395, 402, 409, 414, 421, 431, 439, 442, 447, 481
last, change, after, activated	502
last, List, Void	52
Layout	230, 534, 539
layouts	534
left, accepts	493
left, String, Integer	68
left, String, String	69
left, String, String	74
level	237, 454, 459, 713
LF, BOUNDED	558, 667, 668, 669
LF, SIMPLE	558, 666, 667
limit, Date, Date, Range	131
limit, Date, Range, Date, Range	131
limit, Quantity, Quantity, Range	100
limit, Quantity, Range, Quantity, Range	100
LINEAR	556, 597
linear, quantity	597
LINK	228, 229, 242, 243, 244, 352, 353, 354, 355, 355, 559, 568, 569, 570, 716, 717, 718, 719
List	567, 895, 896
List	52
list, index, List, Void	51
list, price	418, 428
list, number	82
Load	228, 282, 288, 289, 290, 291, 381, 556, 596
load	533
load, data, layout (String)	534
load, files	534
List	367
Load, Plan	229, 379, 381, 382
load, plans	367, 379
load, policy	268, 367, 614, 619
Load, Size	228, 290, 291, 291, 597
load, size, usage	291, 597
load, sized, and, time (Date, Range)	370
load, sized, time (Date, Range)	369
load, sizes	280
load, time (Date, Range)	367
load, time (Date, Range)	367
loading, buffers	274
loading, operations	282
loads	243, 247, 569
List	243, 247, 569
Location	228, 242, 246, 247, 248, 249, 256, 264, 290, 568
location	256, 264
locations	242, 568
lock, allocated	370
lock, ending, on, hand	360
lock, on	382
lock, retain, from, allocated	468
locked, as, planned	378
locks	499
log, Number	82
log, Number, Number	82
log10, Number	83
Log	325, 328, 242, 247, 251, 261, 282, 292, 293, 296, 301, 303, 309, 326, 327, 328, 330, 331, 332, 338, 339, 362, 364, 375, 378, 382, 383, 398, 402, 407, 408, 421, 430, 442, 446, 455, 460, 461, 468, 470, 476, 480, 482, 484, 485, 486, 488, 489, 499, 502, 504, 511, 518, 533, 534, 535, 543, 546, 548, 567, 568, 582, 586, 587, 604, 623, 627, 628, 632, 714, 786, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000
logical, String, Symbol	76

Index

Lot	229, 338, 361, 362, 385
lot	385
List	338, 362, 385
Lot, Flow	229, 362, 384, 385, 386
LOT, NOT CONSUMED	560, 565, 782, 783, 851
LOT, NOT PRODUCED	560, 565, 783, 851, 852
LOT, OVER, CONSUMED	560, 565, 782, 851
LOT, OVER, PRODUCED	560, 565, 783, 851, 852
lot	358, 384
lot, unacted	251
LOW, ON, HAND	560, 565, 784, 852, 853
low, on, hand	780, 781, 782, 784
low, on, hand (Date)	681, 684, 690, 693, 697, 701, 705, 708, 712
low, on, hand, profile	680, 684, 689, 693, 697, 701, 704, 708, 712
lower, String	69
M	
maneuver	267, 370, 611
maneuver, type, Void, Type	151
managed	242, 568
MANUAL	558, 561, 563, 671, 786, 813
MARGIN, JARGEL (Date, Date)	238
margin, count, String, String	75
matching, forecasts	427
max, all	789, 790, 816, 817, 855
max, bucket, load	616
MAX, EFFICIENCY	561, 297
max, deal	500
max, List, Date	116
max, List, Integer	108
max, List, Number	108
max, List, Percentage	109
max, List, Quantity	109
max, List, Time	109
max, operation, count	612
max, price	399, 407
max, quantity, Range	101
max, rank	857
max, resolve, count	500
max, run, time	500
max, size	272, 612
max, stable, resolve, count	500
max, stable, time	500
max, target (Date)	678, 681, 687, 690, 694, 698, 702, 705, 709
max, target, profile	677, 681, 686, 690, 694, 698, 701, 705, 709
MAXIMIZE, PROFIT	561, 566, 794, 860, 861
MAXIMIZE, REVENUE	561, 566, 794, 861
mean, demand, calendar	674
mean, lead, time, calendar	674
Measure	35, 511
measure, quantity	83
measure, String	83
measure, String, Symbol	84
Measure, Unit	36
member	336
member, forecast	435
member, of (Explicit, Slice)	238
member, of (Set)	309
member, of (User)	553
MEMBER, RANK	555, 558, 581, 582, 583, 715
members	236, 309, 350, 449, 533
members, accepted	472
members, allocated	471
members, available	474
members, committed	466
members, consumed	473
members, forecasted	463
members, planned	469
memory, size	154
mid, String, Integer	69
mid, String, Integer, Integer	70
mid, String, String, Integer	70
mid, x	517
mid, y	518
min, all	789, 790, 816, 817, 855
min, bucket, oversize	848
min, cost	504
min, deficit	849, 850, 853
min, delivery, lead, time	317, 324
min, earliness	821, 824, 827, 832, 839, 842
min, excess	822, 823, 828, 833, 841, 843, 853, 854
min, excess, time	852
min, lead	500
min, lateness	820, 823, 826, 831, 839, 841
min, List, Date	116
min, List, Integer	108
min, List, Number	109
min, List, Percentage	110
min, List, Quantity	110
min, List, Time	111
min, load	615
min, load, force	615
min, on, hand	649, 650, 655, 656, 657, 663, 665
min, on, hand, calendar	626, 630
min, overload, std, time	847
min, overload, time	847
min, oversize	847
min, quantity	317, 324
min, quantity, Range	101
min, rank	857
min, shortness	821, 824, 828, 832, 840, 842
min, size	272
min, time	502, 649, 650, 655, 657, 658, 663, 665
min, time, calendar	627, 631
min, underload	848
MINIMIZE, COST	561, 566, 794, 860
MINIMIZE, LATENESS	561, 566, 792, 859
MINIMIZE, PROBLEM, COUNT	561, 566, 792, 858, 859
MINIMIZE, PROBLEMS	561, 566, 792, 858, 859
MINIMIZE, SHORTNESS	561, 566, 793, 794, 860

[illegible]

Index

problem_categories	348, 335, 355, 371, 380, 486, 719	product_items	31	
List	348, 335, 359, 371, 380, 486, 493, 719	Product_Group	228, 320, 327	
Problem_Category	230, 492, 493	product_List_Increase	11	
problem_count	792	product_List_Number	11	
problem_detectors	257, 268, 285	product_List_Percentage	11	
problem_filter	810, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854	Product_List_Quantity	320	
828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854		Product_Root_Zone	317, 318, 319, 320, 323, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854	
problem_selection	487, 500, 790, 791, 818	product_tool	32	
Problem_Sets	220, 498, 502, 503, 504, 563, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854	product_tool_forecast	454	
problem_sets	498	product_tool_forecast	31	
problem_sized_and_time(Diac Range)	371	Product_Supplier	228, 317, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689,	
problem_sized_and_time(Diac Range)	371	Product_Supplier	228, 317, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689,	
problem_sized_and_time(Diac Range)	371	Product_Supplier	228, 317, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689,	
problem_sized_and_time(Diac Range)	371	Product_Supplier	228, 317, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689,	
problem_sized_and_time(Diac Range)	371	Product_Supplier	228, 317, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689,	
problem_sized_and_time(Diac Range)	371	Product_Supplier	228, 317, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393,	

Index

promised_short	407	428
promises	352, 555	716
promising_policy	398, 804, 805, 806	807
promising_policy_map		417
propagation	788, 789	816
proper_list		70
Property_List		37
PROPORTIONAL_INTERACTION	.561, 563, 790, 818	
PROPORTIONAL_RESOLUTION	.561, 563, 790, 817	
QUANTITIES	511	
quantities	566, 567, 862, 868, 875	876
Quantity	31, 248, 249, 272, 297, 298, 299, 317, 324, 330, 334, 355, 341, 343, 354, 357, 360, 361, 363, 369, 370, 371, 377, 384, 385, 407, 409, 428, 429, 430, 434, 436, 446, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 478, 479, 500, 512, 513, 514, 515, 516, 525, 567, 574, 575, 576, 577, 579, 597, 598, 599, 600, 612, 622, 623, 624, 625, 626, 629, 630, 649, 650, 655, 656, 657, 658, 663, 664, 665, 673, 674, 675, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 715, 718, 780, 781, 782, 783, 784, 785, 801, 802, 811, 822, 824, 825, 828, 833, 840, 841, 842, 843, 849, 850, 853, 854, 862, 868, 875, 876, 877, 889, 890, 891, 377, 384, 385, 406, 428, 436, 445, 513, 655, 664, 801,	802, 862
quantity (Date)		361
quantity_integer		88
quantity_interval_Quantity	Quantity	103
quantity_late_power		859
quantity_number		330
quantity_and		310
quantity_per	297, 298, 299, 334, 335, 341, 343, 598, 599, 600	
Quantity_Range	38, 406, 428, 445, 567, 624, 625, 629, 636,	658, 668, 670, 671, 890, 891
quantity_range_Quantity_Quantity	624, 625, 656, 658, 668, 670	
quantity_range_String		103
quantity_range_String_Symbol		104
quantity_range_power		104
quantity_String		88, 89
quantity_String_Symbol		90
quantity_Time		90
quarters.Date_Range		138
quoted		395
quote_end_of_bucket		167
quote_larger_multiple		330
quote_multiple		330
REQUEST_ISSUED		594
request_running		567, 871, 877
REQUEST_NOT_PLANNED	559, 563, 720, 721, 820	
rank		96
rand_integer_integer_integer		9
rand_number_number		9
rand_seed_number		9
rank_231, 262, 297, 308, 324, 335, 375, 399, 418, 510, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 744, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929,		

[illegible]

[illegible][illegible]

Index

week_periods	311
weekday_format	894
WEEKDAYS	566, 863
WEEKENDS	566, 863
WEEKLY	555, 577, 578, 580
WEEKS	562, 810
weeks_date_range	141
weeks_date_range_incr	142
WEIGHTED_LATENCY	561, 566, 793, 859
WEIGHTED_SHORTNESS	561, 566, 793, 860
where	165
wherein	165
while_logical_void	160
while	517, 895
whileend_string_string	76
with_connection_connection_void	153
with_echo_file_string_void	156
within_location	247
within_date_date_range	134
within_date_date_range	135
within_daylight_savings_time_date	125
within_leap_year_date	125
within_quantity_quantity_range	98
within_quantity_range_quantity_range	99
Worksheet	230, 534, 540
workbooks	534
X	
x	517
x_position	248
x_size	248
x_size_min	248
xor_logical_logical	144
Y	
y	517
y_position	248
y_size	249
y_size_min	248
year_date	124
YEARLY	566, 867
yield	599, 600
yield_near	599, 600
Z	
ZERO	557, 610, 611
zero_available_to_promote	476